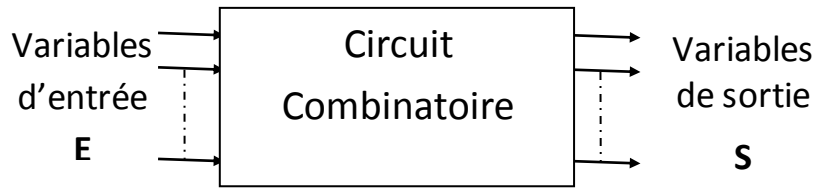
Chapitre 3 / Exemples de circuits combinatoires

Plan :

1. Introduction

1.1 Définition

- Un système est dit combinatoire, lorsque les variables de sortie dépendent uniquement des variables d'entré : $\boldsymbol{S = F(E)}$



- A chaque configuration des entrées correspond une configuration unique des sorties

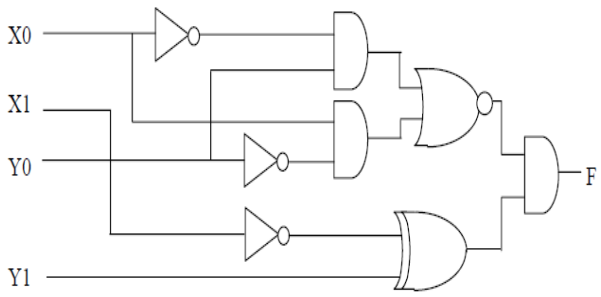
1.2 Analyse d'un circuit Combinatoire

Consiste à trouver sa fonction logique

Méthodologie :

- Donner l'expression des sorties en fonction de ses entrées
- En déduire au final la (ou les) fonction(s) logique(s) du circuit

Exemple



$$\boldsymbol{F = \overline{(\overline{X_0}.Y_0 + X_0.\overline{Y_0})}.(\overline{X_1} \oplus Y_1)}$$
$$\boldsymbol{= (\overline{Y_0 \oplus X_0}).(\overline{X_1 \oplus Y_1})}$$
$$\boldsymbol{= (Y_0 \odot X_0).(Y_1 \odot X_1)}$$

$$\boldsymbol{F = 1 \ si \ X_0 = Y_0 \ et \ X_1 = Y_1}$$

Le circuit est donc un comparateur indiquant l'égalité de deux nombres

$(\boldsymbol{X = X_1\ X_0}\quad \boldsymbol{et}\quad \boldsymbol{Y = Y_1\ Y_0})$.

1.3 Synthèse d'un circuit combinatoire

La synthèse d'un circuit logique combinatoire consiste à proposer un logigramme répondant à un cahier de charge (ou à une table de Vérité), les étapes de conception sont les suivantes:

- Identifier les entrées et les sorties (IN / OUT) du circuit.
- Construire la table (les tables) de vérité.
- Identifier chaque fonction à partir de la table de vérité.
- Simplifier chaque fonction.
- Dessiner le schéma du circuit.

2 Exemples de circuits combinatoires

2.1 Les circuits de transcodage

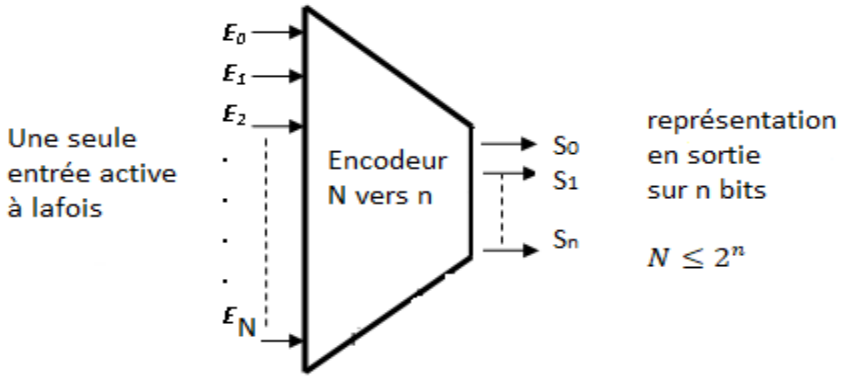
Un circuit de transcodage est un circuit qui transforme une information présente en entrée sous une forme donnée : ***code1,*** en une information équivalente en sortie mais sous une autre forme : ***code 2***

Les transcodeurs sont de 3 types :

- les codeurs (encodeurs)
- les décodeurs
- les transcodeurs

a. Codeurs (Encodeur)

Un codeur est un circuit à ***N*** entrées et ***n*** sorties qui code en binaire le rang de la seule entrée activée.



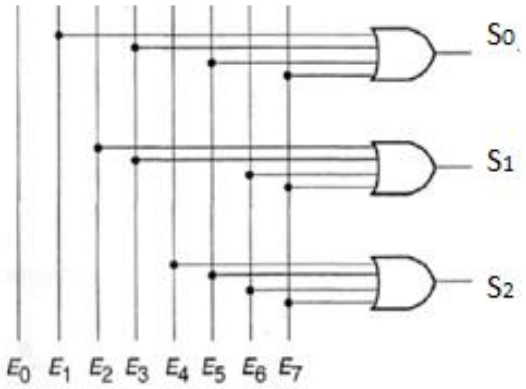
Exemple : Encodeur sur 3 bits $\rightarrow 2^3 = 8$ entrées et 3 sorties

	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	S_2	S_1	S_0
0	1								0	0	0
1		1							0	0	1
2			1						0	1	0
3				1					0	1	1
4					1				1	0	0
5						1			1	0	1
6							1		1	1	0
7								1	1	1	1

$S_2 = E4 + E5 + E6 + E7$

$S_1 = E2 + E3 + E6 + E7$

$S_0 = E1 + E3 + E5 + E7$

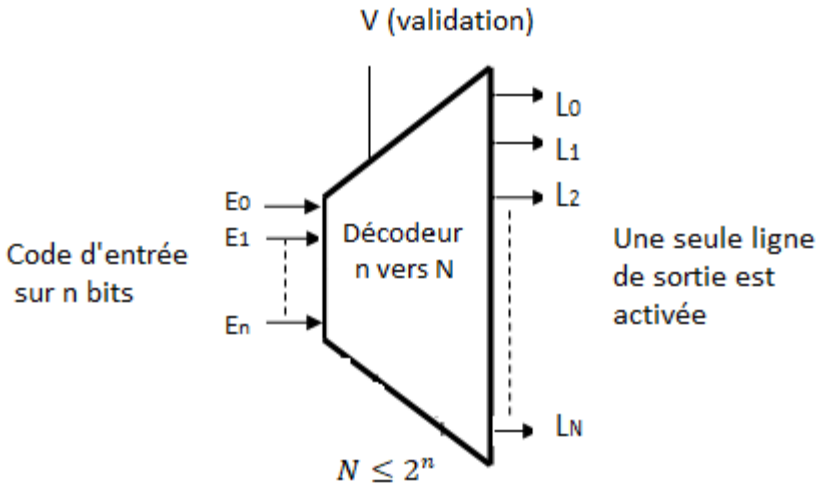


Application pratique : codeurs de clavier numérique

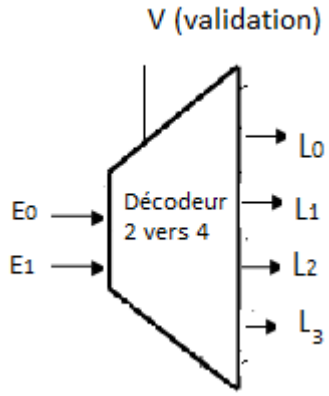
b. Décodeur

Joue le rôle inverse d'un codeur, il est constitué de :

- n entrées de données : Code sur n bits
- 2^n sorties (lignes)
- A chaque code en entrée, une ligne de sortie est sélectionnée



Exemple Décodeur 4 voies (2 vers 4)



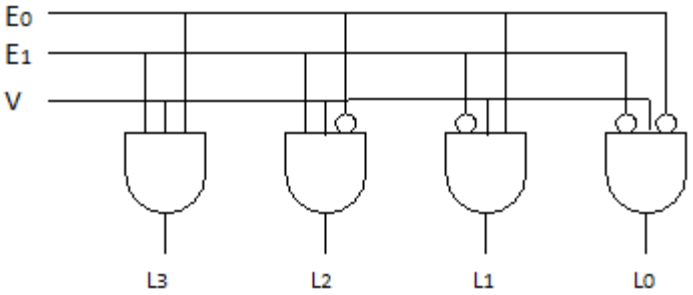
V=: 0

E1	E0	L3	L2	L1	L0
0	0	0	0	0	0
0	1	0	0	0	0
1	0	0	0	0	0
1	1	0	0	0	0

V=1

E1	E0	L3	L2	L1	L0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

$L_0 = \bar{E}_1.\bar{E}_0.V$ $L_1 = \bar{E}_1.E_0.V$ $L_2 = E_1.\bar{E}_0.V$ $L_3 = E_1.E_0.V$



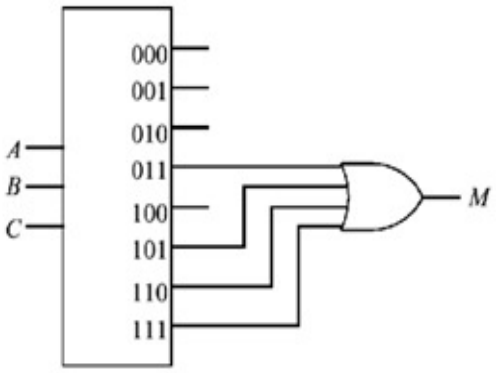
Application pratique :

- Adressage d'une mémoire
- Génération de fonctions

Exemple génération de la fonction Majorité à l'aide d'un décodeur 3 vers 8

$$M = CB\overline{A} + C\overline{B}A + \overline{C}BA + CBA$$

A	B	C	M
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

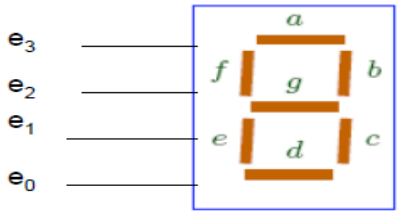


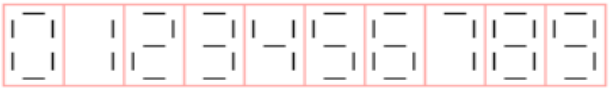
c. *transcodeurs*

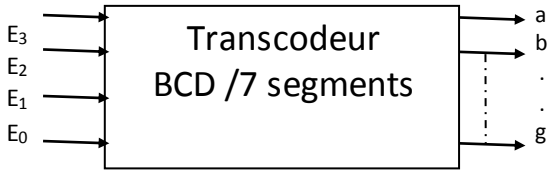
C'est un circuit combinatoire qui permet de transformer un ***code X*** (sur n bits) en entrée, en un ***code Y*** (sur m bits) en sortie

Exemple : Transcodeur BCD / afficheur 7 segments

On veut afficher les 10 chiffres décimaux à l'aide de 7 segments, notés de a à g, qui peuvent être à 0 (éteint) ou 1 (allumé). Le codage des 10 chiffres décimaux nécessite 4 bits, que l'on peut noter ***E_3 à E_0.***







- Table de vérité du transcodeur ***BCD/ 7*** segments.

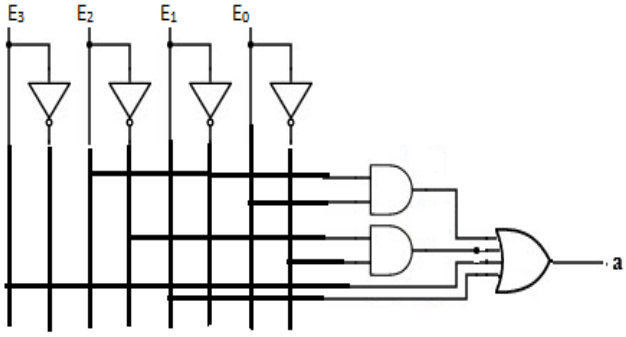
	E_3	E_3	E_3	E_3		a	b	c	d	e	f	g
0	0	0	0	0		1	1	1	1	1	1	0
1	0	0	0	1		0	1	1	0	0	0	0
2	0	0	1	0		1	1	0	1	1	0	1
3	0	0	1	1		1	1	1	1	0	0	1
4	0	1	0	0		0	1	1	0	0	1	1
5	0	1	0	1		1	0	1	1	0	1	1
6	0	1	1	0		1	0	1	1	1	1	1
7	0	1	1	1		1	1	1	0	0	0	0
8	1	0	0	0		1	1	1	1	1	1	1
9	1	0	0	1		1	1	1	0	0	1	1
	1	0	1	0		X	X	X	X	X	X	X
	1	0	1	1		X	X	X	X	X	X	X
	1	1	0	0		X	X	X	X	X	X	X
	1	1	0	1		X	X	X	X	X	X	X
	1	1	1	0		X	X	X	X	X	X	X
	1	1	1	1		X	X	X	X	X	X	X

- Expression simplifiée de la sortie ***a***

E_3E_2 / E_1E_0	00	01	11	10
00	1	0	X	1
01	0	1	X	1
11	1	1	X	X
10	1	1	X	X

$$a = E_3 + E_1 + E_2E_0 + \overline{E_2}.\overline{E_0}$$

- Proposition d'un logigramme pour la sortie ***a***

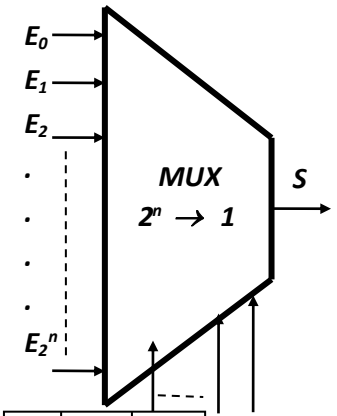


On peut procéder de la même manière pour les autres sorties

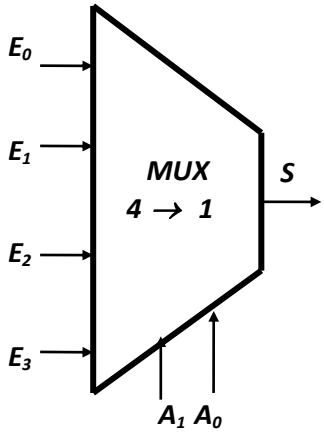
2.2 Circuits d'aiguillage

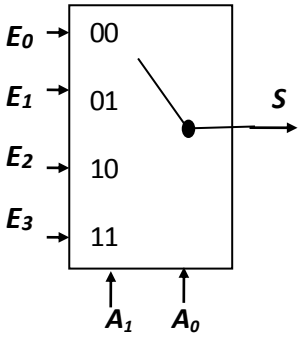
a. Le multiplexeur (sélecteur de données).

Le multiplexeur est un circuit combinatoire Sélecteur qui possède 2^n entrées d'information, n entrées de commande (Adresse), et une seule sortie. Son rôle consiste à sélectionner, à l'aide de signaux de commande, une des entrées et à l'envoyer à la sortie



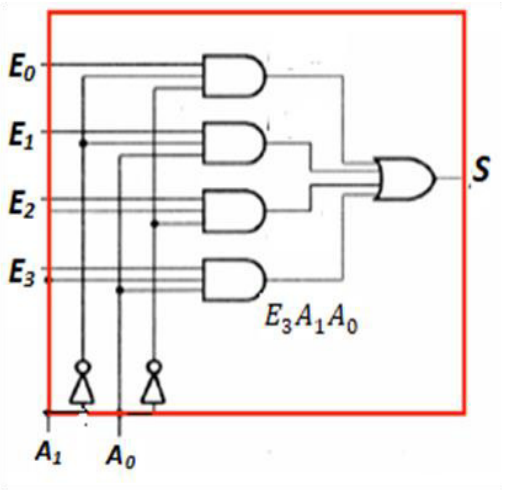
Exemple : N=2





A_1	A_0	S
0	0	E_0
0	1	E_1
1	0	E_2
1	1	E_3

$$S = E_0\overline{A}_1.\overline{A}_0 + E_1\overline{A}_1A_0 + E_2A_1.\overline{A}_0 + E_3A_1A_0$$



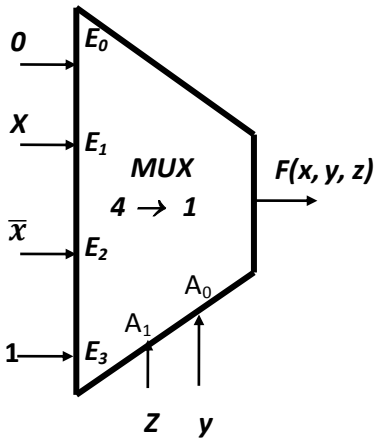
Les multiplexeurs ont de nombreuses applications. Ils peuvent par exemple être utilisés comme :

- sélecteur de données.
- convertisseur parallèle-série. Le multiplexeur reçoit en parallèle des données qu'il peut transmettre l'une après l'autre sur sa sortie.
- générateur de fonctions logiques.
  En fait un multiplexeur à n entrées d'adresses (et donc 2^n entrées de données) peut réaliser toutes les fonctions logiques combinatoires de $n+1$ variables.

Exemple : réalisation de la fonction $F = x \bullet y \bullet \bar{z} + \bar{x} \bullet \bar{y} \bullet z + y \bullet z$ avec un multiplexeur à 2 entrées d'adresse.

On connecte ***2*** des variables aux entrées d'adresse ou de commande. Par exemple z à A_1 et y à A_0. On doit ensuite connecter convenablement les 4 entrées de données de façon à reproduire les différents termes de la fonction ***F***.

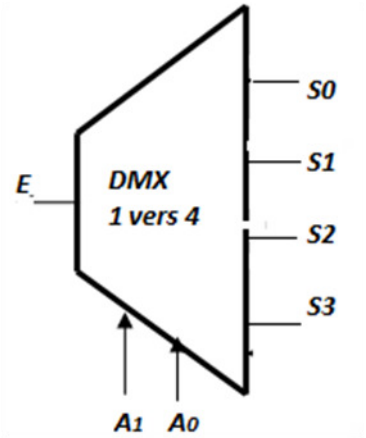
- Pour le terme $x \bullet y \bullet \bar{z}$ on doit relier l'entrée E_1 dont l'adresse est ***01*** ($z = 0,\ y = 1$) à **x**.
- Pour le terme $\bar{x} \bullet \bar{y} \bullet z$ on doit relier l'entrée E_2 dont l'adresse est 10 ($z = 1, y = 0$) à $\bar{x}$.
- Pour le terme $z \bullet y$ on relie l'entrée E_3 dont l'adresse est 11 ($z = 1, y = 1$) au niveau logique **1** .
- Toutes les autres entrées sont connectées au niveau logique **0** (la masse) puisque $F = 0$ pour les valeurs de ***y*** et ***z*** correspondantes.

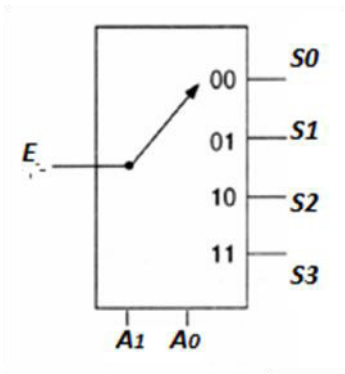


b. démultiplexeur.

Un démultiplexeur joue le rôle inverse du multiplexeur. Il permet de faire passer une entrée de donnée, dans l'une des sorties selon les valeurs des entrées de commandes.
Il possède :

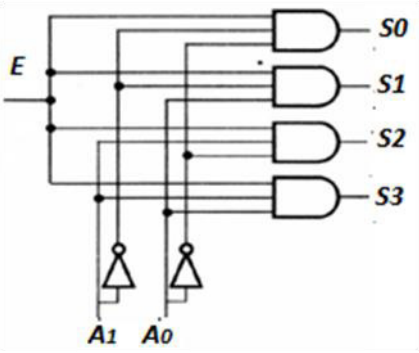
- une entée unique d'information (donnée)
- ***n*** entrées de commandes (adresses).
- ***m*** sorties ($m \leq 2^n$).





A_1	A_0	S_3	S_2	S_1	S_0
0	0	0	0	0	E
0	1	0	0	E	0
1	0	0	E	0	0
1	1	E	0	0	0

$$S_0 = E.\bar{A}_1.\bar{A}_0$$
$$S_1 = E.\bar{A}_1 A_0$$
$$S_2 = E.A_1.\bar{A}_0$$
$$S_3 = E.A_1 A_0$$



Application pratique : Transformation série parallèle.

Remarque

La plupart des multiplexeurs/démultiplexeurs sont munis d'une entrée supplémentaire : entée de validation permettant d'activer ou non le circuit. Lors de la désactivation, les sorties peuvent être placées soit à l'état inactif ("0") soit dans un état dit "haute impédance".

2.3 Comparateur

(Voir TD)

2.4 Additionneur

Pour réaliser des additions binaire de 2 nombres **A** et **B** de n bits on décompose un circuit en deux parties

- Un circuit correspondant à l'addition des bits de poids faible a_0 **et** b_0 (il n'y a pas de retenue propagée à prendre en compte) → ***demi additionneur (Half Adder)***
- Un circuit correspondant à l'addition des bits de poids supérieur a_i **et** b_i (prendre en compte la retenue r_{i-1} propagée depuis le rang ***i-1*** inférieur) → ***additionneur complet (Full Adder)***

a. Demi-additionneur

a_0	b_0	Som	R
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

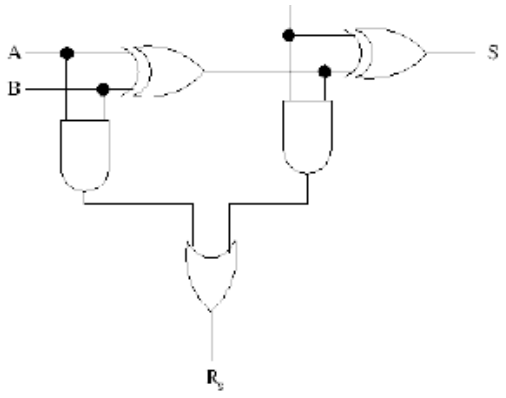
$Som = a_0 \oplus b_0 \quad R = a_0 b_0$

Le circuit logique associé est donc constitué de deux portes, un OU exclusif pour le résultat et un ET pour la retenue:

b. additionneur complet

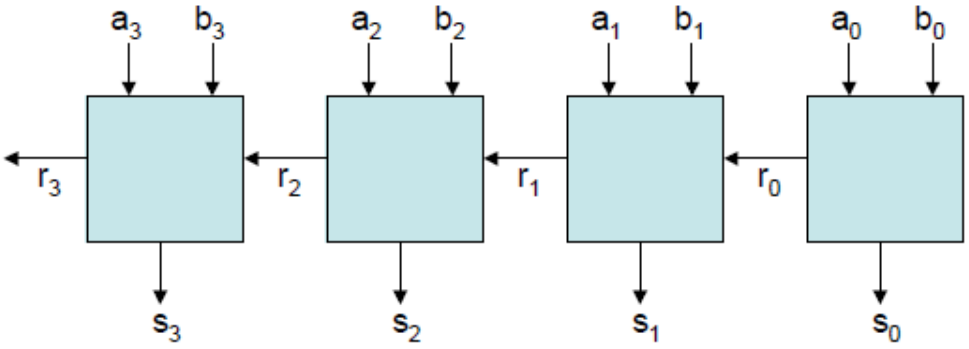
Afin de permettre une liaison de plusieurs additionneurs en série, un additionneur doit avoir une retenue en entrée Re en plus de la retenue en sortie Rs:

A	B	R_e	S	R_s
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



c. Additionneur complet n bits

- L'additionneur **n** bits est obtenu en chaînant entre eux un demi-additionneur et **n-1** additionneurs 1 bit complets
- Le chaînage s'effectue par le biais des retenues propagées



Additionneur
4 bits