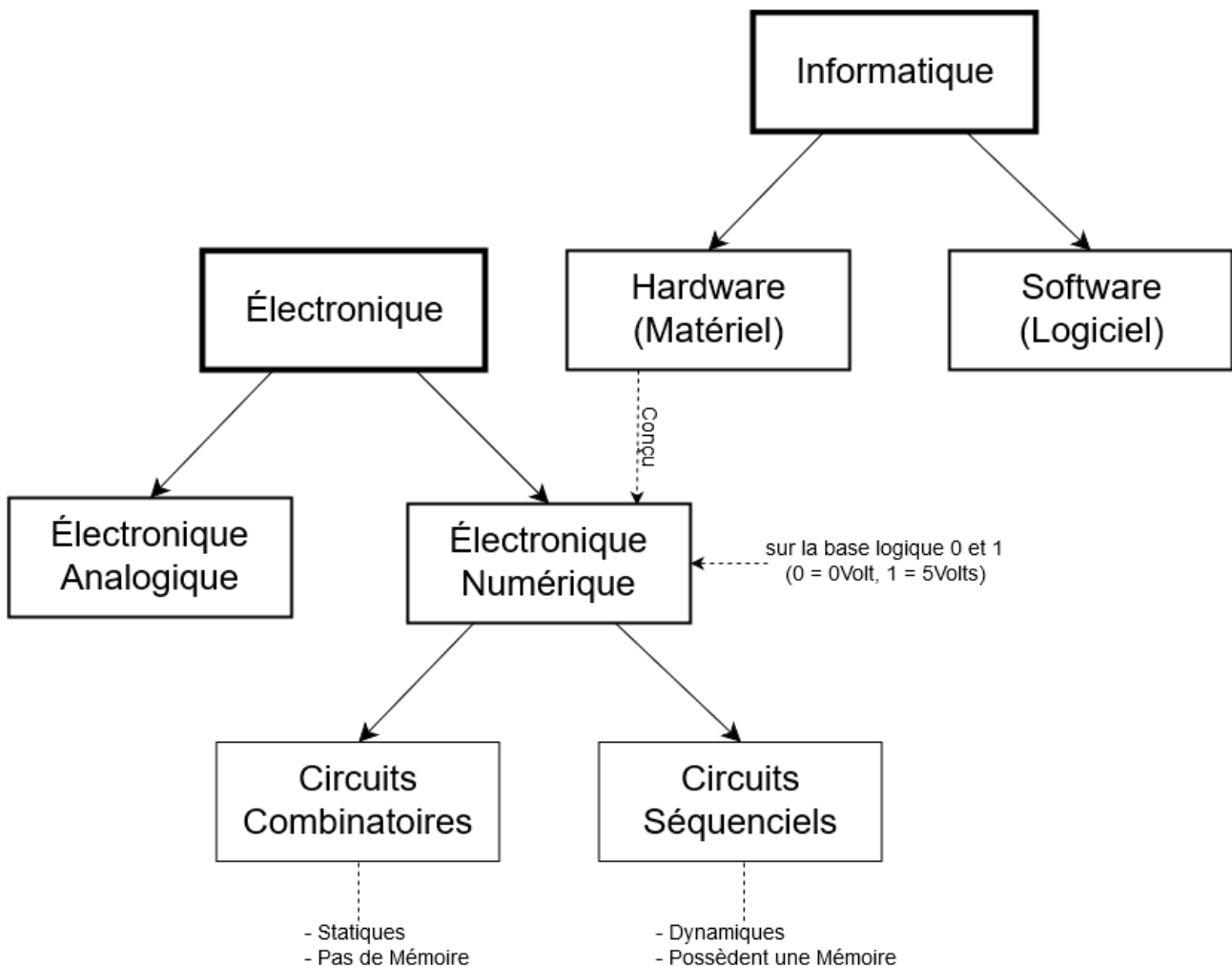


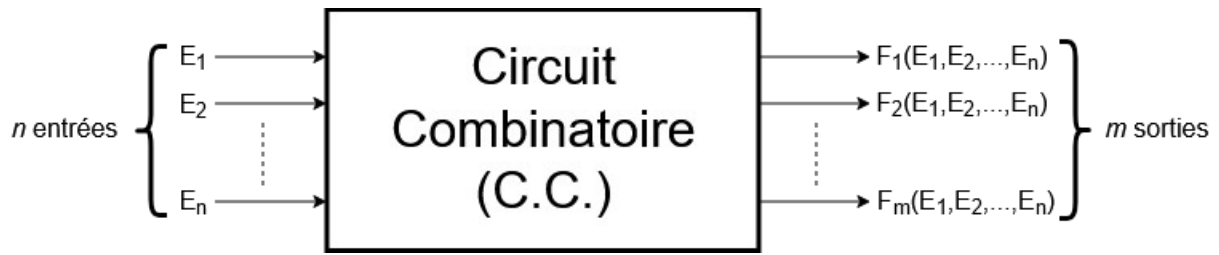
Chapitre 1 : Les Circuits Combinatoires

1. Introduction



2. Définition

- Circuit électronique numérique qui fait à l'intérieur un traitement binaire des entrées vers les sorties.
- Possède n entrées et m sorties binaires numériques discrètes (0/1).
- Sa spécification est à base de Fonctions Booléennes et/ou de Table de Vérité.





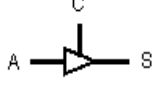


3. Les Portes Logiques (P.L.)

- Composant élémentaire (indivisible).
- Élément basique physique (réel) de l'électronique numérique.
- Ils sont assemblés en composition pour former les circuits numériques.
- Ils représentent une opération élémentaire de la logique Booléenne (et, ou, non...).

Les différentes portes sont :

Porte	Symbole	Table de vérité	Description															
AND		<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>S</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	A	B	S	0	0	0	0	1	0	1	0	0	1	1	1	C'est le et logique : $S = A \cdot B$
A	B	S																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
OR		<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>S</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	A	B	S	0	0	0	0	1	1	1	0	1	1	1	1	C'est le ou logique : $S = A + B$
A	B	S																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
NOT		<table border="1"> <thead> <tr> <th>A</th> <th>S</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> </tr> </tbody> </table>	A	S	0	1	1	0	C'est le non logique : $S = \bar{A}$									
A	S																	
0	1																	
1	0																	
NAND		<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>S</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	A	B	S	0	0	1	0	1	1	1	0	1	1	1	0	C'est le non-et (NOT-AND) logique : $S = \overline{A \cdot B}$
A	B	S																
0	0	1																
0	1	1																
1	0	1																
1	1	0																

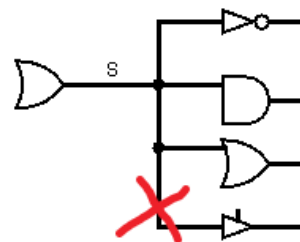
NOR		<table border="1"> <thead> <tr><th>A</th><th>B</th><th>S</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	A	B	S	0	0	1	0	1	0	1	0	0	1	1	0	C'est le non-ou (NOT-OR) logique : $S = \overline{A+B}$
A	B	S																
0	0	1																
0	1	0																
1	0	0																
1	1	0																
XOR		<table border="1"> <thead> <tr><th>A</th><th>B</th><th>S</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	A	B	S	0	0	0	0	1	1	1	0	1	1	1	0	C'est le ou-exclusif logique : $S = A \oplus B$
A	B	S																
0	0	0																
0	1	1																
1	0	1																
1	1	0																
XNOR		<table border="1"> <thead> <tr><th>A</th><th>B</th><th>S</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </tbody> </table>	A	B	S	0	0	1	0	1	0	1	0	0	1	1	1	C'est le non-ou-exclusif logique : $S = \overline{A \oplus B} = A \otimes B$
A	B	S																
0	0	1																
0	1	0																
1	0	0																
1	1	1																
buffer		<table border="1"> <thead> <tr><th>A</th><th>S</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td></tr> </tbody> </table>	A	S	0	0	1	1	C'est le buffer , il ne fait aucune opération logique, il est utilisé pour réduire la vitesse d'un signal dans certaines situations: $S = A$									
A	S																	
0	0																	
1	1																	
tristate buffer		<table border="1"> <thead> <tr><th>A</th><th>C</th><th>S</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>Z</td></tr> <tr><td>1</td><td>0</td><td>Z</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </tbody> </table>	A	C	S	0	0	Z	1	0	Z	0	1	0	1	1	1	C'est le buffer à 3 états , ça permet d'utiliser le signal logique Z : $\begin{cases} si(C=0) \Rightarrow S=Z \\ si(C=1) \Rightarrow S=A \end{cases}$
A	C	S																
0	0	Z																
1	0	Z																
0	1	0																
1	1	1																

(A,B sont les entrées. S c'est la sortie. C c'est la commande)

Fan-out : Caractéristique de la porte qui indique le nombre maximum que peut fournir sa sortie S pour les entrées des portes suivantes.

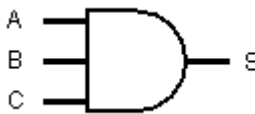
Exemple : fan-out = 3

S ne peut pas fournir à plus de 3 sorties.



Fan-in : Caractéristique de la porte qui indique le nombre d'entrées d'une porte

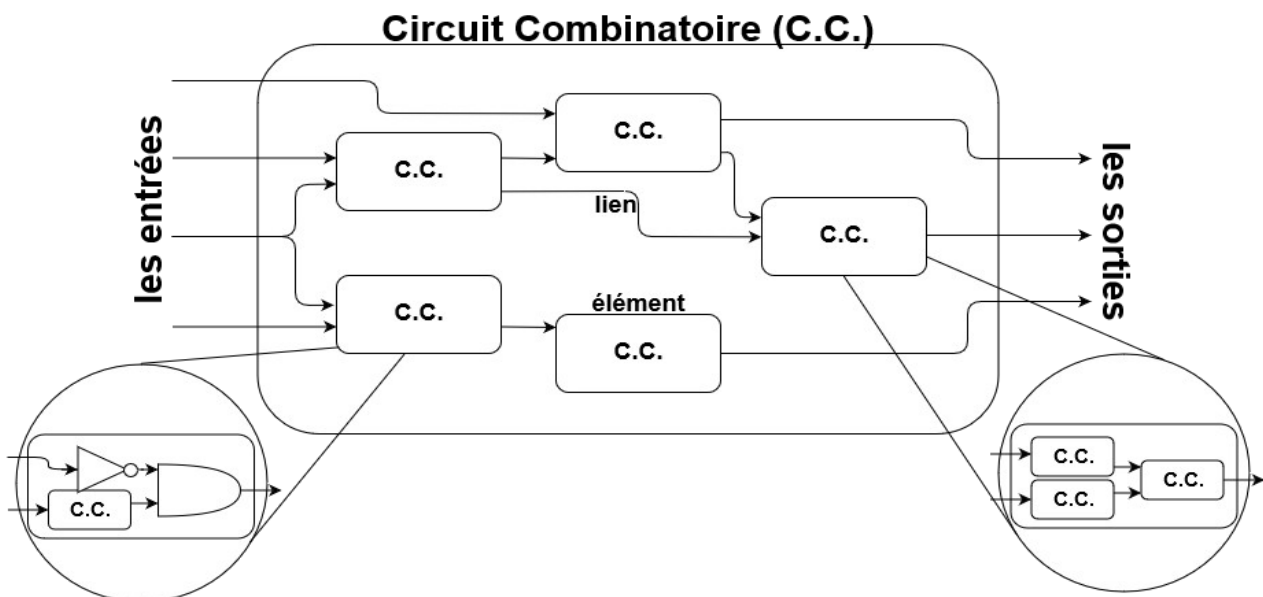
Exemple : La porte **AND3** est une porte *AND* avec 3 entrées.

Porte	Symbole	Table de vérité																																				
AND3		<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>C</th> <th>S</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td><td>1</td></tr> </tbody> </table>	A	B	C	S	0	0	0	0	0	0	1	0	0	1	0	0	0	1	1	0	1	0	0	0	1	0	1	0	1	1	0	0	1	1	1	1
		A	B	C	S																																	
		0	0	0	0																																	
		0	0	1	0																																	
		0	1	0	0																																	
		0	1	1	0																																	
		1	0	0	0																																	
		1	0	1	0																																	
		1	1	0	0																																	
		1	1	1	1																																	

Remarque : Le fan-out peut être rajouté comme nombre à la fin du nom d'une porte (ex : AND3), et omis si c'est le nombre par défaut d'entrées(ex :AND = AND2).

4. Règles de construction d'un circuit combinatoire

un circuit combinatoire est décrit formellement comme un **graphe**, constitué d'un ensemble d'**éléments**, interconnectés par des **liens** orientés.



Un circuit combinatoire doit aussi respecter les 5 règles suivantes pour qu'il soit valide :

- L'*élément* ne peut être qu'un circuit combinatoire ou bien une porte logique. Cette règle met en valeur le caractère d'emboîtement hiérarchique des circuits complexes.
- Le *lien* est un fil électrique transportant 0 ou 5 Volts (0 ou 1 logique). Il existe trois types ; les entrées, les sorties, et liens internes.
- Pour chaque combinaison d'entrée, il n'existe qu'une seule unique combinaison de sortie, mathématiquement parlant c'est une fonction (ou application). C'est aussi l'origine du nom *combinatoire*.
- L'entrée d'un élément ne peut pas recevoir son signal de plus d'une seule sortie précédente, le cas échéant induit à un court-circuit (ou contention).
- Le chemin d'un signal ne peut pas traverser un élément plus d'une fois. Autrement dit, il n'y a pas de cycle (boucle) dans le circuit.

Si l'une des règles n'est pas respectée, le circuit n'est pas un circuit combinatoire.

5. Spécification d'un circuit combinatoire

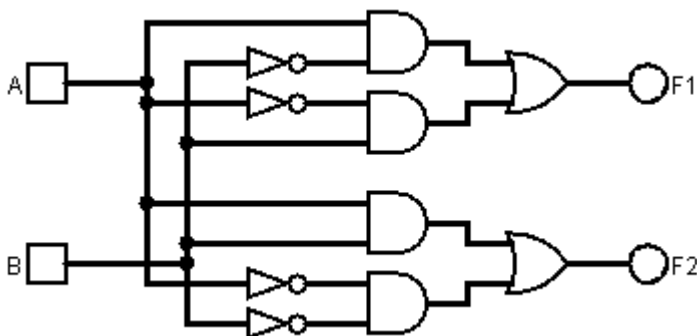
La spécification (description de fonctionnement) d'un circuit combinatoire est assurée par deux outils formels qui sont ; les Fonctions Booléennes et/ou la Table de Vérité.

Exemple :

$$F_1(A,B) = A \cdot \bar{B} + \bar{A} \cdot B$$

$$F_2(A,B) = A \cdot B + \bar{A} \cdot \bar{B}$$

Solution :



A	B	F ₁	F ₂
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

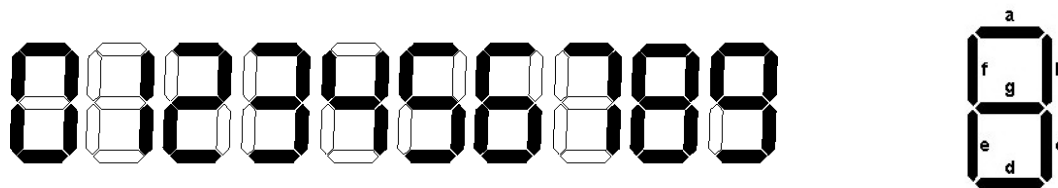
6. Étapes conventionnelles de construction d'un circuit combinatoire

Conventionnellement 5 étapes sont suivies pour construire un circuit combinatoire :

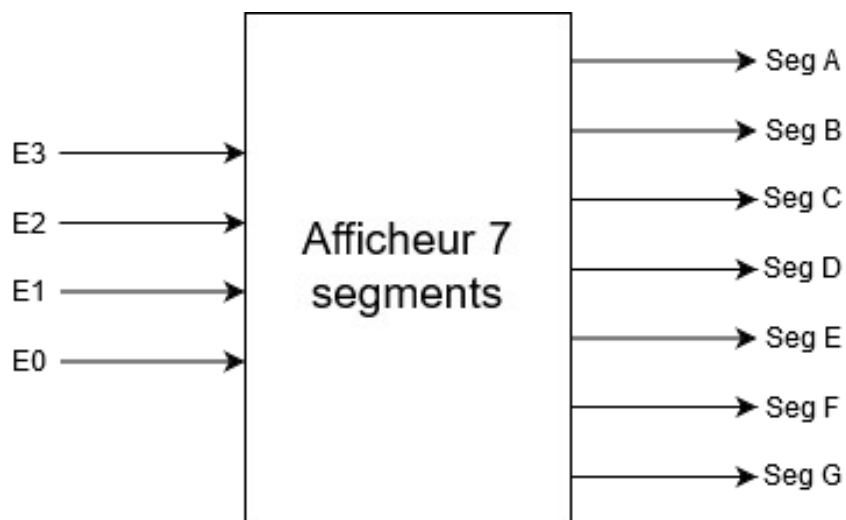
- Schéma global
- Table de Vérité
- Fonctions Booléennes
- Table de Karnaugh ou simplification Algébrique
- Logigramme

Exemple :

Le circuit combinatoire d'un afficheur 7 segments est un circuit qui commande un afficheur 7 segments, il permet l'affichage d'un seul chiffre sur 7 segments comme illustré sur le schéma, il reçoit en entrée un nombre encodé sur 4 bits appartenant à l'intervalle [0,9] et en sortie il produit la combinaison des segments qui affiche le chiffre dans le système décimal.



Étape 1 : Schéma global



Étape 2 : Table de Vérité

E3	E2	E1	E0	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	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

Étape 3 : Fonctions Canoniques Disjonctives

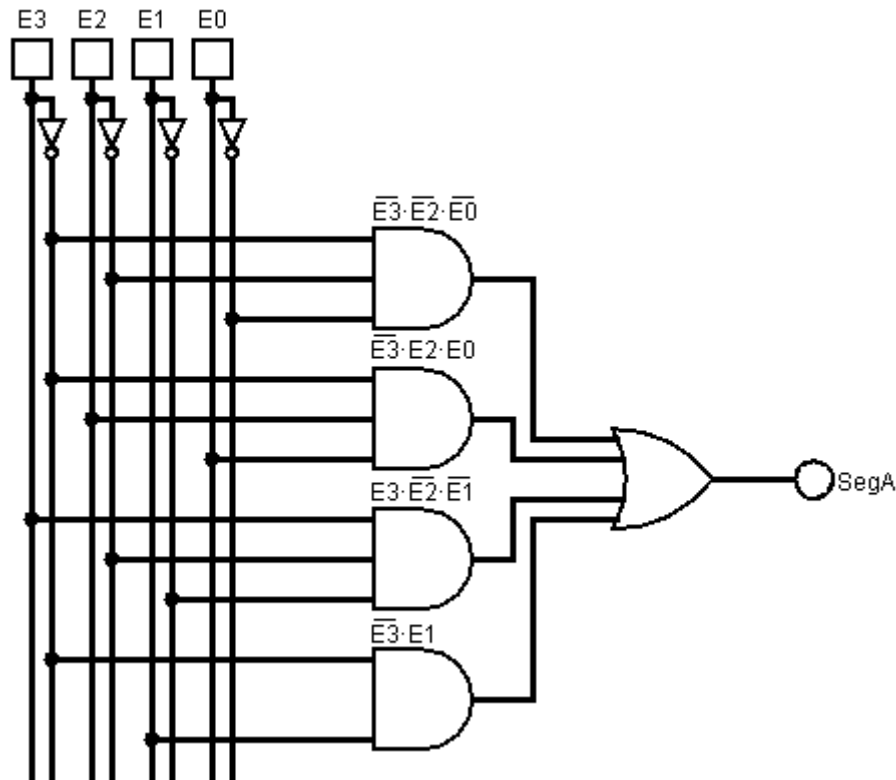
$$A(E3,E2,E1,E0) = \overline{E3} \cdot \overline{E2} \cdot \overline{E1} \cdot \overline{E0} + \overline{E3} \cdot \overline{E2} \cdot E1 \cdot \overline{E0} + \overline{E3} \cdot \overline{E2} \cdot E1 \cdot E0 + \overline{E3} \cdot E2 \cdot \overline{E1} \cdot E0 + \overline{E3} \cdot E2 \cdot E1 \cdot \overline{E0} + \overline{E3} \cdot E2 \cdot E1 \cdot E0 + E3 \cdot \overline{E2} \cdot \overline{E1} \cdot \overline{E0} + E3 \cdot \overline{E2} \cdot E1 \cdot E0$$

Étape 4 : Tables de Karnaugh

		E3E2			
		00	01	11	10
E1E0	00	1	0	0	1
	01	0	1	0	1
	11	1	1	0	0
	10	1	1	0	0

$$A(E3,E2,E1,E0) = \overline{E3} \cdot \overline{E2} \cdot \overline{E0} + \overline{E3} \cdot E2 \cdot E0 + E3 \cdot \overline{E2} \cdot \overline{E1} + \overline{E3} \cdot E1$$

Étape 5 : Logigramme



Remarque 1: Cette méthode est appelée *Méthode à 5 étapes*, ou *Circuit à 2 étages*, en raison que sur le schéma il y a 2 étages (étage des AND et étage du OR).

Remarque 2: Il est aussi possible de travailler avec la FCC (Forme Canonique Conjonctive) sur les étapes 3, 4 et 5, c'est plus optimal si le nombre de 0 est inférieur aux 1 dans la sortie (colonne A), sinon c'est la FCD la plus optimal. L'optimalité ici est dans la réduction du nombre de termes.

Remarque 3: Les principaux mécanismes de réduction d'un circuit sont; de choisir entre la FCD et la FCC en rapport avec le nombre de 0 et 1 dans la sortie, La Table de Karnaugh ou la Simplification Algébrique et enfin les Sorties Indéfinies.

7. Les sorties indéfinies (don't care)

Dans certains cas la sortie peut être indéfinie dans la spécification d'un circuit, par exemple dans l'afficheur 7 segments les chiffres de 10 à 15 ne sont pas définis, et cette caractéristique peut être utilisée comme avantage pour réduire plus le nombre de portes dans le circuit. Dans la Table de Karnaugh, les sorties indéfinies sont prises ou-bien 0 ou-bien 1 selon la situation qui permet de réduire le plus le circuit.

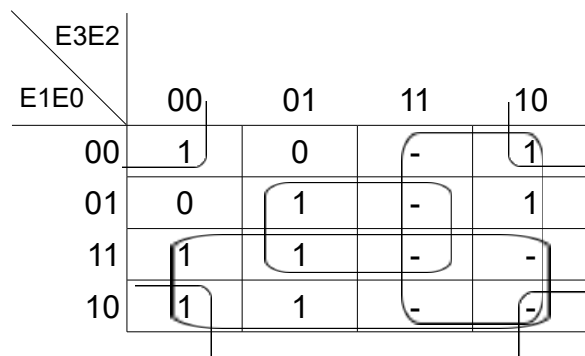
Étape 2 : Table de Vérité

E3	E2	E1	E0	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	-	-	-	-	-	-	-
1	0	1	1	-	-	-	-	-	-	-
1	1	0	0	-	-	-	-	-	-	-
1	1	0	1	-	-	-	-	-	-	-
1	1	1	0	-	-	-	-	-	-	-
1	1	1	1	-	-	-	-	-	-	-

Étape 3 : Fonctions Canoniques Disjonctives

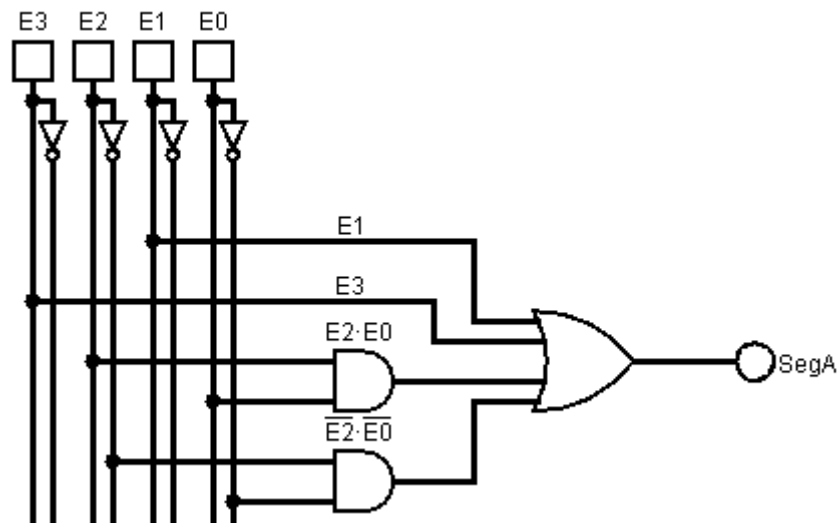
$$A(E3,E2,E1,E0) = \overline{E3} \cdot \overline{E2} \cdot \overline{E1} \cdot \overline{E0} + \overline{E3} \cdot \overline{E2} \cdot E1 \cdot \overline{E0} + \overline{E3} \cdot \overline{E2} \cdot E1 \cdot E0 + \overline{E3} \cdot E2 \cdot \overline{E1} \cdot E0 + \overline{E3} \cdot E2 \cdot E1 \cdot \overline{E0} + \overline{E3} \cdot E2 \cdot E1 \cdot E0 + E3 \cdot \overline{E2} \cdot \overline{E1} \cdot \overline{E0} + E3 \cdot \overline{E2} \cdot E1 \cdot E0$$

Étape 4 : Tables de Karnaugh



$$A(E3,E2,E1,E0) = E1 + E3 + E2 \cdot E0 + \overline{E2} \cdot \overline{E0}$$

Étape 5 : Logigramme



Remarque 1: La réduction dans le nombre des portes est très important, ça permet de réduire le coût du circuit, de le rendre plus rapide, de réduire sa consommation électrique, et de simplifier sa réalisation physique.

Remarque 2: Il ne faut pas confondre les très de sorties indéfinies, avec les très de réduction, qui sont eux appliqués sur les entrées de la Table de Vérité.

très de réduction : Contrairement aux sorties indéfinies, les très de réduction non pas d'implication sur la spécification du circuit, ils sont utilisés juste pour réduire la taille de la Table de Vérité, lorsque plusieurs lignes partagent la même sortie. Un très de réduction signifie quelle que soit la valeur de la variable en entrée.

Exemple :

A	B	C	S1	S2
0	0	0	1	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	0
1	0	1	0	1
1	1	0	0	0
1	1	1	1	0

↔

A	B	C	S1	S2
0	-	-	1	0
1	0	0	0	0
1	0	1	0	1
1	1	0	0	0
1	1	1	1	0

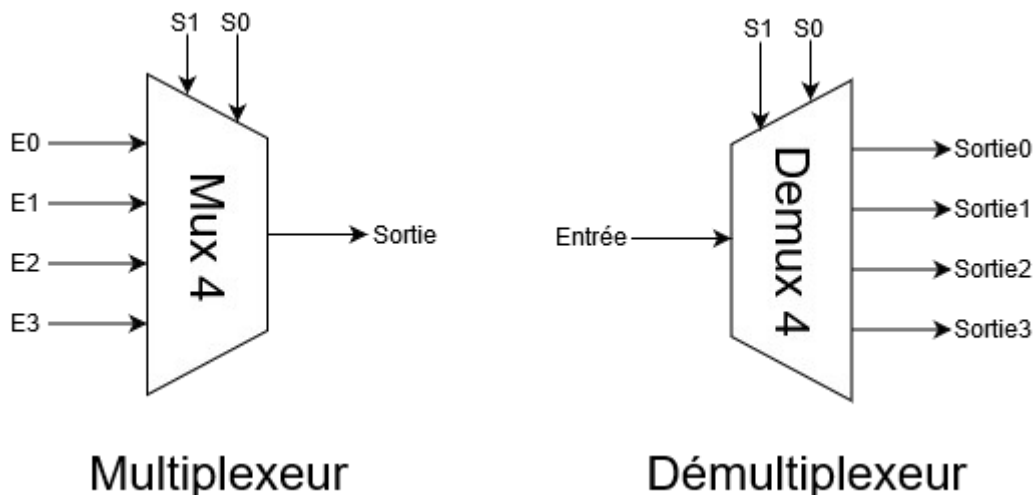
8. Les circuits combinatoires les plus connus

Dans cette section on va décrire les circuits combinatoires les plus communs.

8.1. Multiplexeur/Démultiplexeur

- Le Multiplexeur (ou Sélecteur) est un circuit combinatoire qui comporte 2^n entrées et n sélecteurs avec une seule sortie.
- Son fonctionnement est de sélectionner à travers les sélecteurs (S1 et S0 dans l'exemple) une seule entrée et de la faire passer vers la sortie.
- L'entrée à faire passer est la valeur binaire encodée dans les sélecteurs, les autres entrées sont bloquées.
- Le Multiplexeur est souvent désigné comme le *if* ou le *switch* du Hardware.

Dans le schéma suivant on a un multiplexeur avec 4 entrée, 2 sélectionneurs et une sortie.



- Le démultiplexeur fait l'opération inverse.
- Il a une seule entrée et 2^n sorties et n sélecteurs.
- Il fait traverser l'entrée vers la sortie désignée par les sélecteurs.

Le Multiplexeur et le Démultiplexeur ont un rôle primordial pour la mise en œuvre des bus de communication entre différents blocs hardware, et dans la gestion et le contrôle du flux de l'information dans les systèmes hardware complexes.

Remarque 1: Le symbole du trapézoïde et la nomenclature Mux/Demux sont communément utilisés dans la littérature de la conception Hardware pour désigner le Multiplexeur/Démultiplexeur.

Remarque 2: Le nombre 4 à la fin du nom Mux et Demux indique le nombre d'entrée dans le Multiplexeur et de sorties dans le Démultiplexeur.

8.2. Encodeur/Décodeur

- L'Encodeur est un circuit avec 2^n entrées et n sorties.
- L'Encodeur doit recevoir qu'une seule entrée à 1, les autres doivent être à 0. En sortie il retourne la valeur encodée en binaire de la position de cette sortie.
- Sur le schéma, si E2 est à 1 et les autres doivent être à 0, la sortie est 10 (2 en binaire).

Sur le schéma $n = 2$ pour l'Encodeur et pour le Décodeur.



- Le Décodeur est l'inverse, il a n entrées et 2^n sorties.
- Le fonctionnement c'est aussi l'inverse, il reçoit une entrée encodée en binaire, et n'active qu'une seule sortie, celle en position avec la valeur encodée en entrée.
- Sur le schéma, si E1=1 et E0=0 (2 en binaire) c'est S2 qui est à 1 et les autres à 0.

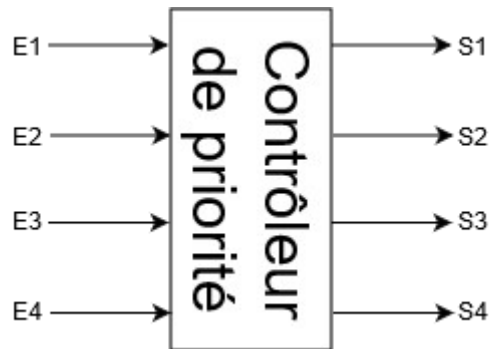
Remarque 1: Les deux nombres dans la nomenclature de l'Encodeur et du Décodeur (4 et 2 sur le schéma) sont les nombres des entrées-sorties des deux composants.

Remarque 2: Le décodeur est un circuit primordial pour l'implémentation des mémoires (RAM/ROM).

8.3. Contrôleur de Priorité

- ➔ Le Contrôleur de Priorité est un circuit combinatoire qui comporte n entrées et n sorties.
- ➔ Son rôle est de construire une échelle de priorité sur les entrées, et de ne faire sortir que l'entrée prioritaire.
- ➔ Il ne fait sortir qu'une seule sortie, celle qui coïncide avec la plus grande priorité parmi les entrées.
- ➔ Sur l'exemple du schéma, l'échelle de priorité est de E1 en descendant à E4. Donc si par exemple E2 et E4 sont actives (mis à 1), la sortie est S2, parce que E2 est plus prioritaire que E4 sur l'échelle.

Dans le schéma suivant le Contrôleur de Priorité est à $n = 4$. Donc 4 entrées, et 4 sorties.

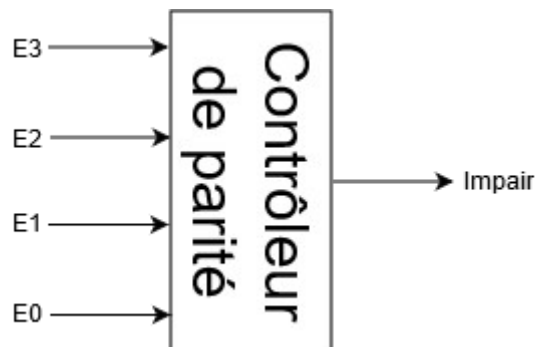


Remarque : On peut trouver ce circuit dans u gestionnaire des *interruptions* par exemple.

8.4. Contrôleur de parité

- Le Contrôleur de Parité est un circuit combinatoire qui comporte n entrées et 1 sortie.
- Il permet de reconnaître la parité des entrées. Ça veut dire, de distinguer si le nombre de 1 dans les entrées est pair ou impair.
- Si le nombre de 1 dans les entrées est impair il fait sortir 1, sinon si c'est pair il fait sortir 0.
- Sur l'exemple, le Contrôleur de Parité a 4 entrées, si l'entrée est par exemple 1110 il va sortir 1 (impair) en raison que le nombre de 1 est 3.

Sur le schéma le Contrôleur de Parité à 4 entrées et une sortie.

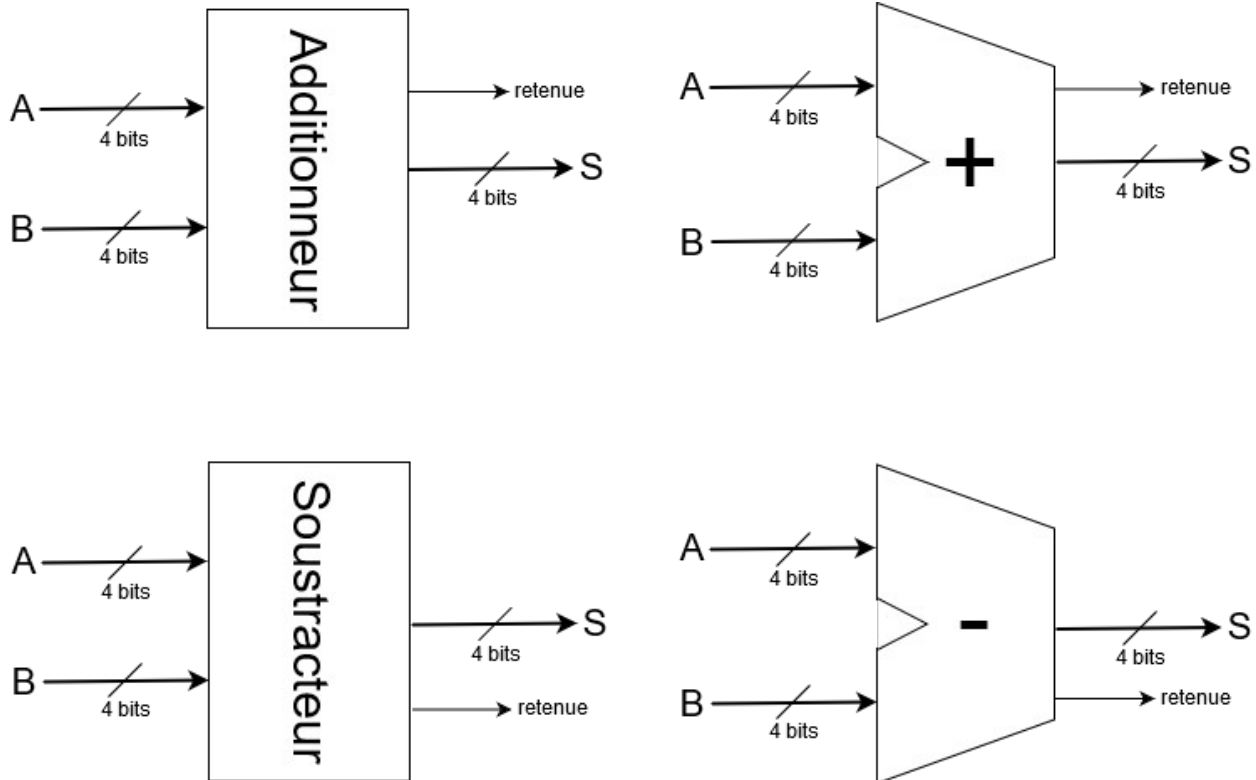


Remarque : Le Contrôleur de Parité est utilisé généralement pour vérifier l'intégrité des données dans les réseaux de transmission.

8.5. Additionneur/Soustracteur

- L'Additionneur est un circuit combinatoire qui fait l'addition binaire entre 2 nombres A et B sur n bits, et donne en sortie la somme sur n bits , plus le bit de la retenue.
- Le Soustracteur est un circuit combinatoire qui fait la soustraction entre 2 nombres A et B sur n bits, et donne en sortie le résultat sur n bits, plus le bit de la retenue.

Sur le schéma on a un Additionneur sur 4 bits avec 2 représentations possibles et un Soustracteur sur 4 bits avec 2 représentations aussi.



Remarque 1: La deuxième représentation de l'Additionneur et du Soustracteur, celle du trapézoïde avec une encoche pour séparer les 2 entrées, est bien commune dans la littérature de la conception Hardware, elle est généralement utilisée pour représenter les opérations arithmétiques et souvent aussi l'UAL.

Remarque 2: Le trait en slash (/) sur les entrées et la sortie avec la mention *4 bits*, indique que par exemple l'entrée A est constituée de 4 entrées différentes (A3, A2, A1, A0). Cette représentation est utilisée pour simplifier la schématisation des *bus* contenant plusieurs fils.

Remarque 3: Le bit de retenue est le même que celui restant en dernier dans les opérations d'additions ou de soustraction binaire.

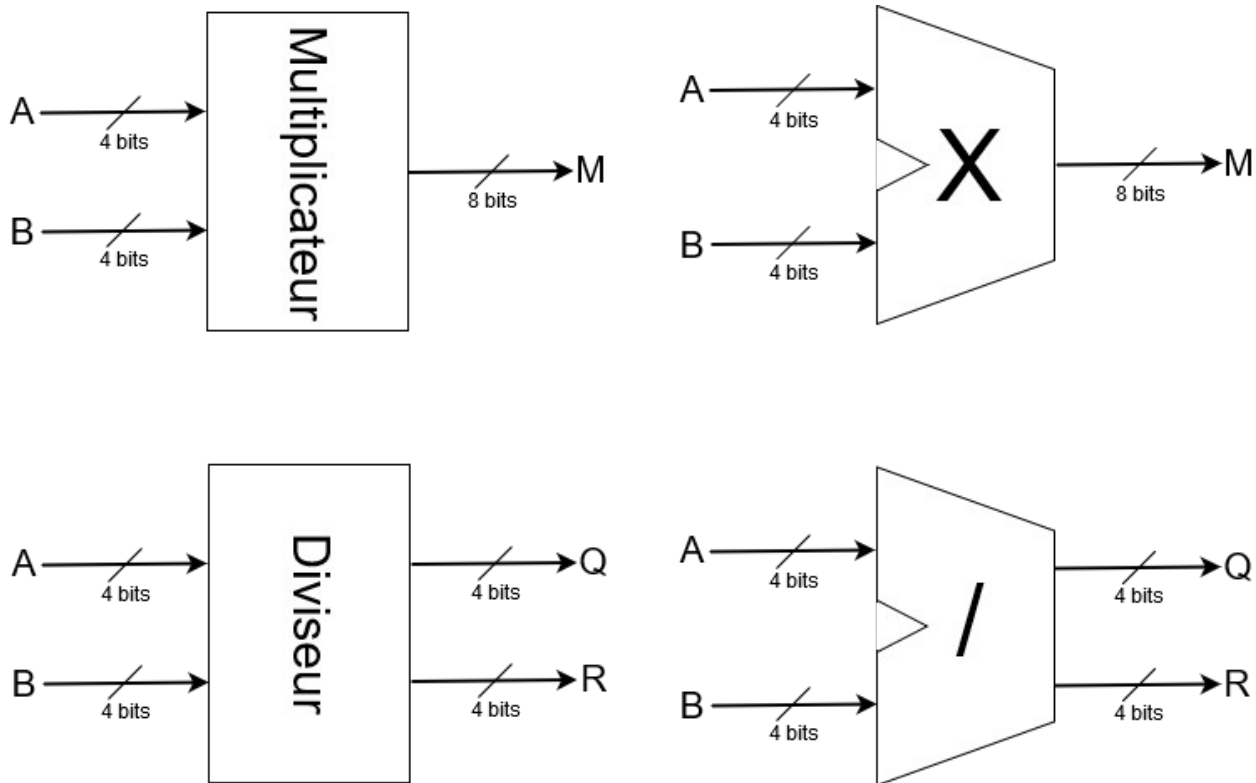
Remarque 4: Il existe une implémentation différente de l'Additionneur et du Soustracteur selon l'encodage des valeurs opérées. Ça veut dire qu'il y a des versions différentes de l'additionneur pour l'encodage *Non-Signé*, *Signé et Valeur Absolue*, *Complément-à-1*, et *Complément-à-2*.

Remarque 5: Vous allez le plus souvent trouver les circuits qui font les opérations arithmétiques dans les calculatrices et les UAL.

8.6. Multiplicateur/Diviseur

- Le Multiplicateur est un circuit combinatoire qui fait la multiplication binaire entre 2 *Entiers Non Signés* A et B sur n bits, et donne en sortie le résultat de la multiplication sur $2n$ bits.
- La sortie du Multiplicateur doit impérativement être sur $2n$ bits, pour pouvoir contenir le résultat de la multiplication.
- Le Diviseur est un circuit combinatoire qui fait la division de 2 *Entiers Non Signés* A et B sur n bits, et produit un résultat sur 2 sorties avec n bits chacune, Qui sont le Quotient et le Reste.

Le schéma représente un Multiplicateur sur 4 bits avec ses 2 représentations possibles et un Diviseur sur 4 bits avec ses 2 représentations aussi.



Remarque 1: Pour des raisons de simplicité le Multiplicateur et le Diviseur utilisent l'encodage *Entier Non Signé*, le plus souvent un circuit complémentaire assez simple est rajouté pour implémenter les encodages signés.

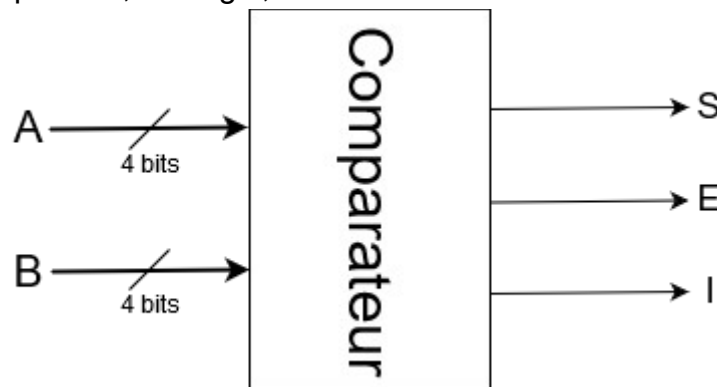
Remarque 2: Le Diviseur ici fait la division entière, avec le quotient et le reste. La division réelle est faite sur les nombres réels (encodage avec virgule flottante IEEE754).

Remarque 3: Il existe aussi une autre implémentation du Multiplicateur et du Diviseur en utilisant les circuits séquentiels, elles possèdent leurs propres avantages et inconvénients.

8.7. Comparateur

- Le Comparateur est un circuit combinatoire qui fait la comparaison entre 2 nombres A et B sur n bits, et produit 3 sorties; Supérieur, Égal et Inférieur.
- Quelques soient les 2 nombres, il n'existe qu'une sortie active possible, Supérieur, Égal ou Inférieur.

Le schéma représente un comparateur avec 4 bits sur les entrées, et les trois sorties de comparaison S = Supérieur, E = Égal, I = Inférieur.



Remarque 1: Le Comparateur fournit les opérations $>$, $=$ et $<$. Pour se procurer \geq , \leq et \neq , il faut rajouter respectivement des portes logiques aux sorties; $S+E$, $I+E$ et \bar{E} .

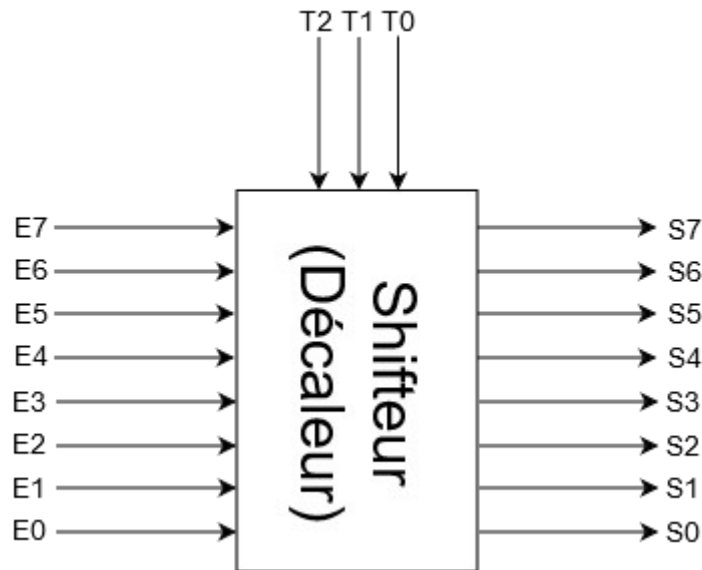
Remarque 2: Le Comparateur comme l'Additionneur et le Soustracteur est relative à l'encodage utilisé pour les nombres comparés (*Non-Signé, Signé et Valeur Absolue, Complément-à-1, Complément-à-2*).

Remarque 3: Pour réduire le nombre des portes logiques et ainsi du Hardware, il n'est pas rare de trouver des micro-architectures sans comparateur. Il est implémenté indirectement par le soustracteur, le résultat de la soustraction est testé s'il est positif, négatif, ou égal à zéro, et interprété comme une comparaison entre 2 nombres.

8.8. Shifteur (Décaleur)

- Le Shifteur est un circuit combinatoire qui fait le décalage à gauche ou à droite des bits en entrées. Le nombre de bits décalés est exprimés binaires par l'entrée Taux. Taux est généralement à n bits lorsque les entrées sont au nombre de 2^n .
- Si par exemple l'entrée dans le schéma en bas est 11001100, le décalage à gauche avec un Taux = 3 (011), ça donnerait 01100000. Tous les bits sont décalés à gauche et les 3 bits les plus à droite sont perdus.
- On a pris ici l'exemple d'un Shifteur à gauche, le Shifteur à droite fait le décalage dans le sens inverse, de gauche vers la droite.
- Le Taux de décalage est encodé en binaire, ici 3 bits pour représenter un décalage de 0 à 7 cases (bits). 7 étant la valeur maximale pour décaler tous les entrées.

Le schéma représente un shifteur à gauche avec 8 bits d'entrées, et 8 bits en sorties, avec un Taux sur 3 bits.



Remarque 1: Le Shiftieur à gauche représente mathématiquement une multiplication de la valeur entrée par 2^{Taux} , et à droite c'est une division par 2^{Taux} .

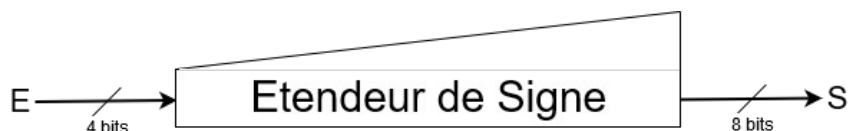
Remarque 2: Il existe plusieurs variantes des Shiftieurs; les Shiftieurs Logiques, les Shiftieurs Arithmétiques et les Shiftieurs à Rotation. Le Shiftieur étudié ici est un Shiftieur dit Logique, dans lequel les bits vides après décalage sont toujours remplis par 0.

Remarque 3: Il existe aussi une implémentation pour le Shiftieur à base de circuits Séquentiels, chacune des implémentations a ses avantages et ses inconvénients.

8.9. Étendeur de Signe

- L'Étendeur de Signe est un circuit combinatoire qui a en entrée une valeur signée sur n bits, et en sortie il va étendre cette valeur sur m bits (tel que $m > n$) en conservant le signe.
- Par exemple sur le schéma, la valeur en entrée $3 = 0011$ sur 4 bits est étendue en $3 = 00000011$ sur 8 bits, ou $-3 = 1101$ en $-3 = 11111101$ en complément-à-2.

Le schéma représente un Étendeur de Signe avec 4 bits en entrées, et 8 bits en sorties, en encodage complément-à-2.



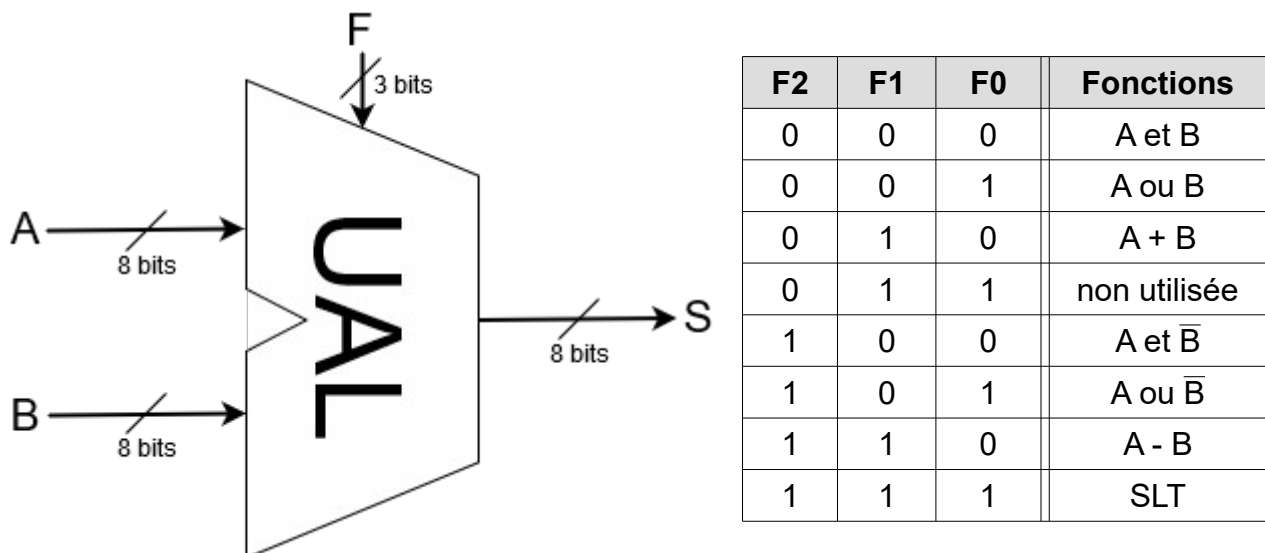
Remarque 1: L'Étendeur de Signe concerne l'encodage des nombres signés (*Signe et Valeur Absolue, Complément-à-1, Complément-à-2*).

Remarque 2: Les Étendeurs sont souvent représentés par la forme géométrique trapèze.

8.10. UAL (Unité Arithmétique et Logique)

- L'UAL est l'unité dans un processeur qui effectue les opérations arithmétiques et logiques.
- L'UAL est un circuit combinatoire qui fait une Fonction (ou Opération) sur ses deux entrées A et B, et produit le résultat sur la sortie S. La fonction est choisie par l'entrée F.
- La table à droite donne le code en F de l'opération à faire par l'UAL.
- Si par exemple en complément-à-2 on donne sur A la valeur 3 = 00000011 et sur B la valeur -2 = 11111110, et on choisit la fonction 2 sur F (010) qui fait A + B, le résultat dans S serait S = 00000001.

Le schéma représente une UAL sur 8 bits utilisant l'encodage complément-à-2, avec 2 entrées A et B sur 8 bits, une sortie S sur 8 bits et la fonction F sur 3 bits.



Remarque 1: Beaucoup de circuits combinatoires impliquent le choix de l'encodage (*Non-Signé, Signé et Valeur Absolue, Complément-à-1, Complément-à-2*), mais la quasi-totalité du Hardware actuel utilise le *Complément-à-2*. De ce fait à partir de ce point, ce cours ne concerne que seulement l'encodage Complément-à-2.

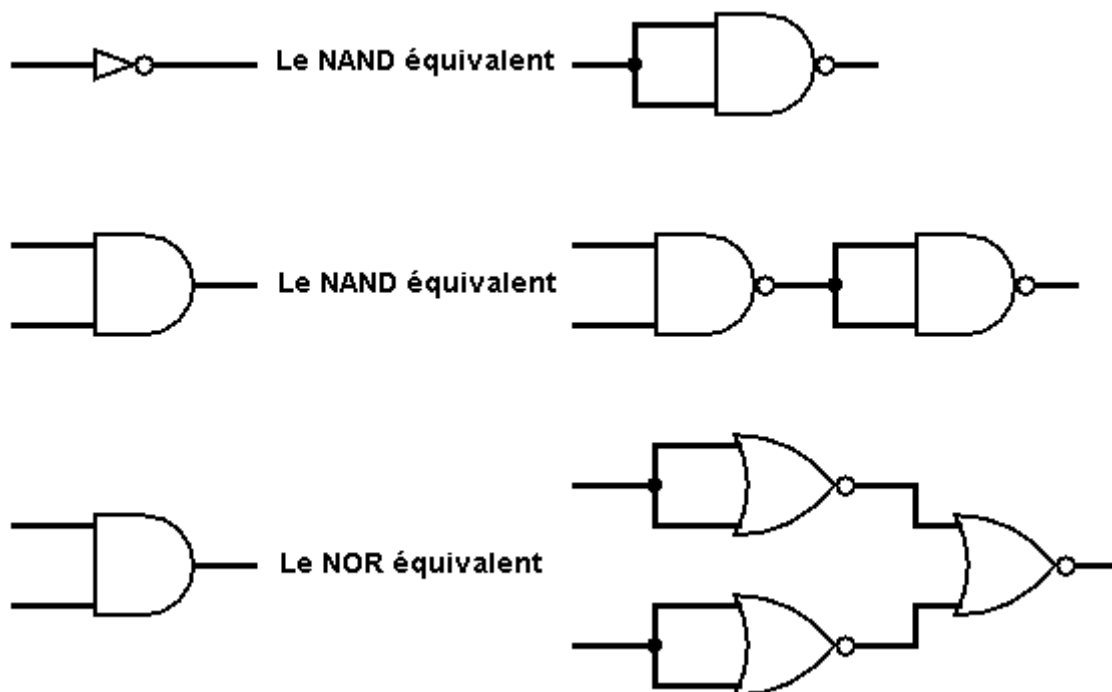
Remarque 2: L'UAL présentée sur le schéma est directement prise du livre *Digital Design and Computer Architecture* (page : 249), elle a servi à la réalisation partielle du processeur MIPS.

Remarque 3: Il est possible de trouver l'UCC (Unité de Commande et de Contrôle) implémentée en circuit combinatoire, sur des processeurs simples du type monocycle comme Arduino ou les Microcontrôleurs PIC et AVR. Mais dans la majeure partie l'UCC est implémenté par un circuit séquentiel programmable évolué.

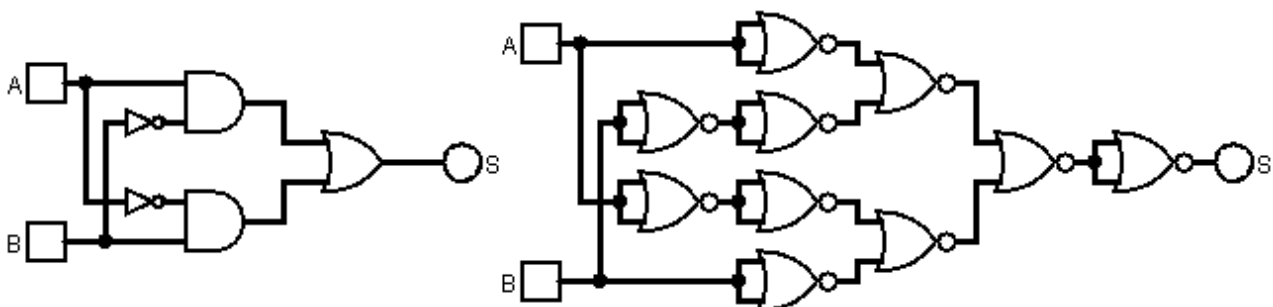
9. Les portes universelles NAND et NOR

- Les portes universelles NAND et NOR sont des portes qui peuvent implémenter n'importe quelle opération booléenne logique (et, ou, non...).
- N'importe quel circuit peut être converti à un circuit qu'avec des portes NAND ou NOR.
- L'implémentation sur silicium (silicon) est favorable aux portes NAND et NOR, car elles requièrent moins de transistors que les autres portes et technologiquement elles sont plus adaptées.
- Une grande partie des circuits intégrés (Processeurs, GPU, Chipset...) sont implémentés à base de NAND ou de NOR sur silicium.

Exemple 1:

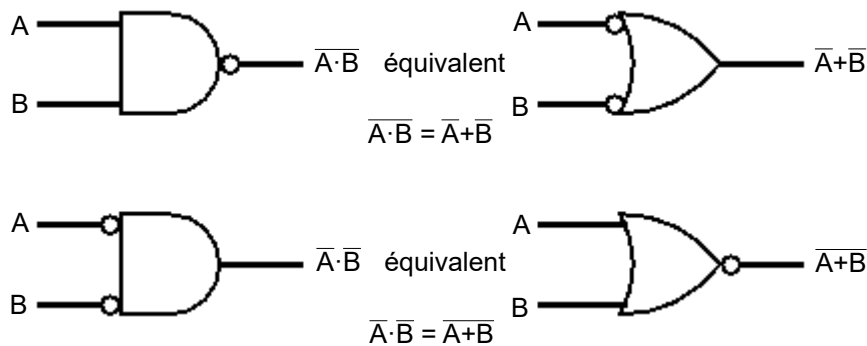


Exemple 2: Convertir le circuit suivant porte par porte en portes NOR.



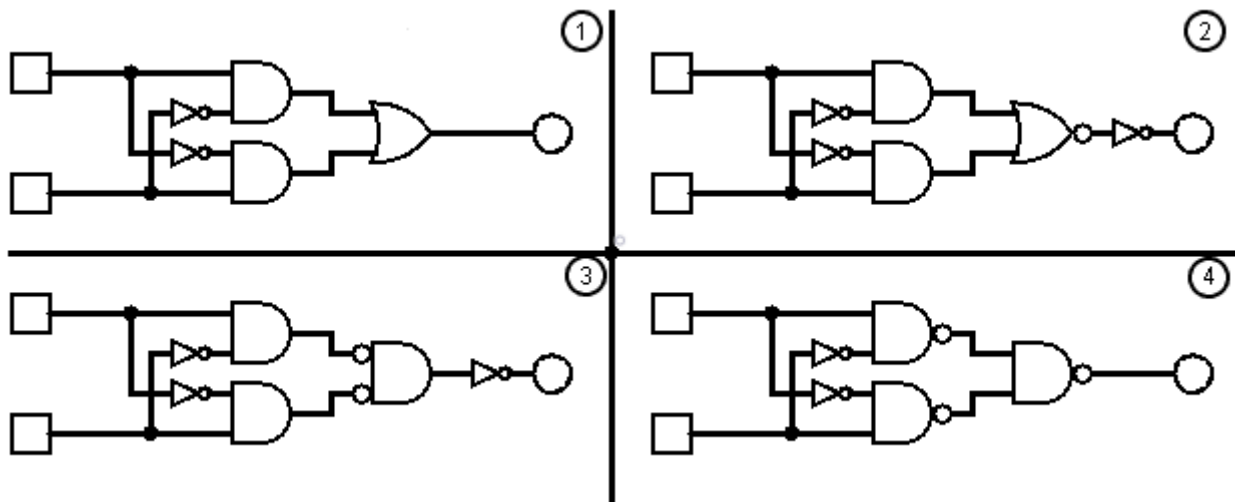
La technique du poussé de bulles :

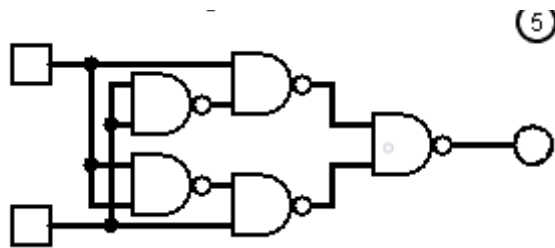
- C'est une technique visuelle graphique se basant sur le théorème de De Morgane, elle permet de transformer avec le minimum de portes n'importe quel circuit en circuit à base de portes NAND ou NOR.
- Cette technique est illustrée sur le schéma en bas $\overline{A \cdot B} = \overline{A} + \overline{B}$ et $\overline{\overline{A} \cdot \overline{B}} = \overline{\overline{A+B}}$.
- Visuellement, pour la première formule $\overline{A \cdot B} = \overline{A} + \overline{B}$, on pourrait imaginer que si lorsqu'on pousse vers l'intérieur la bulle en sortie, elle ressort sur toutes les entrées de la porte, et la porte se transforme en OR.
- Et la deuxième $\overline{\overline{A} \cdot \overline{B}} = \overline{\overline{A+B}}$, si on poussait toutes les bulles en entrées, une bulle ressort en sortie de la porte, et la porte se transforme de AND en OR.
- Les bulles c'est des NOT. 2 NOT sur le même fil s'annulent selon le postulat $\overline{\overline{A}} = A$.



- La règle en général, c'est que si on poussait les bulles sur toutes les entrées, une bulle va ressortir en sortie et la porte se transforme de AND à OR ou inversement.
- Et si on poussais la bulle de la sortie à l'intérieur de la porte, une bulle va ressortir sur chaque entrée et la porte se transforme de AND à OR ou inversement.

Exemple :

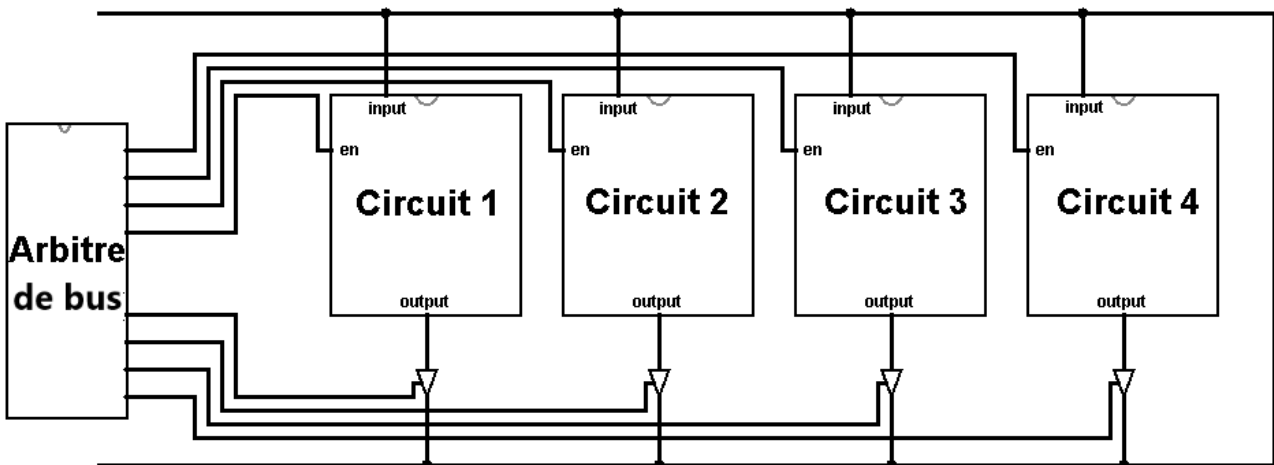




10. La valeur Z et son utilisation dans les Bus

- La valeur Z ou l'état Z est le troisième état logique en comptant l'état 0 et l'état 1.
- La valeur Z est parfois appelée valeur Flottante ou électroniquement valeur à Haute Impédance.
- Un signal (fil) avec une valeur Z signifie que le fil n'est pas branché, ni à 0 ni à 1.
- La seule porte qui puisse générer la valeur Z c'est le Tristate Buffer, si le Tristate Buffer est mis à 0 sur son entrée de commande il débranche le fil de sortie.
- Électriquement, si un fil est débranché son voltage est flottant entre le 0 Volt et le 5 Volts. Expérimentalement c'est un voltage totalement aléatoire.
- Le rôle de la valeur Z est de détourner la règle 4 de la définition des circuits combinatoire, de telle-sorte que plusieurs sorties peuvent partager le même fil.
- Sa principale utilisation c'est le partage d'un fil commun pour toutes les sorties dans le but de transporter l'information. Le fil partagé est généralement appelé Bus.

Le schéma suivant représente un Bus utilisé pour la transmission de données entre 4 circuits. La transmission est gérée par l'Arbitre du Bus.



- Les circuits peuvent être combinatoires ou séquentiels.
- L'Arbitre est le circuit responsable de la gestion du transfert de données entre circuits sur le Bus.
- Chaque sortie (output) d'un circuit est contrôlée par un Tristate Buffer lui-même géré par l'Arbitre de façon que seulement une seule sortie prend le Bus.

- Tous les circuits reçoivent le signal du Bus sur leurs entrées (input), mais le signal ne peut entrer que si l'entrée *en* (enable) du circuit est active (mis-à-1).
- En contrôlant le *enable* des circuits, l'Arbitre gère le circuit qui doit prendre la valeur sur le Bus.
- Si par exemple le Circuit 2 veut transmettre vers le Circuit 4, l'Arbitre doit activer le Tristate Buffer de la sortie du Circuit 2 et le *enable* du circuit 4.

Remarque 1: La confusion la plus connue des novices en électronique numérique est de penser que si un fil est débranché son voltage doit être à 0 volt, donc c'est la valeur logique 0, mais pour rappel, il y a une différence entre le voltage et l'ampérage, c'est vrai que lorsqu'un fil est débranché il n'y a pas de courant qui passe, mais pour l'électronique numérique le signal est transféré par la différence de potentiel et pas par le courant.

Remarque 2: Ce qui implique de la remarque précédente, c'est que la valeur logique 0 doit être impérativement branché électriquement au pôle (-) (appelé aussi la masse, ground ou GND) et le 1 logique doit être branché au pôle (+)(appelé aussi Vcc).

Remarque 3: Pour simplifier la compréhension, l'illustration précédente du Bus est faite sur un seul fil, mais généralement dans le cas réel, un Bus comporte plusieurs fils pour transporter l'information sur 8, 16, 32 bits...etc.

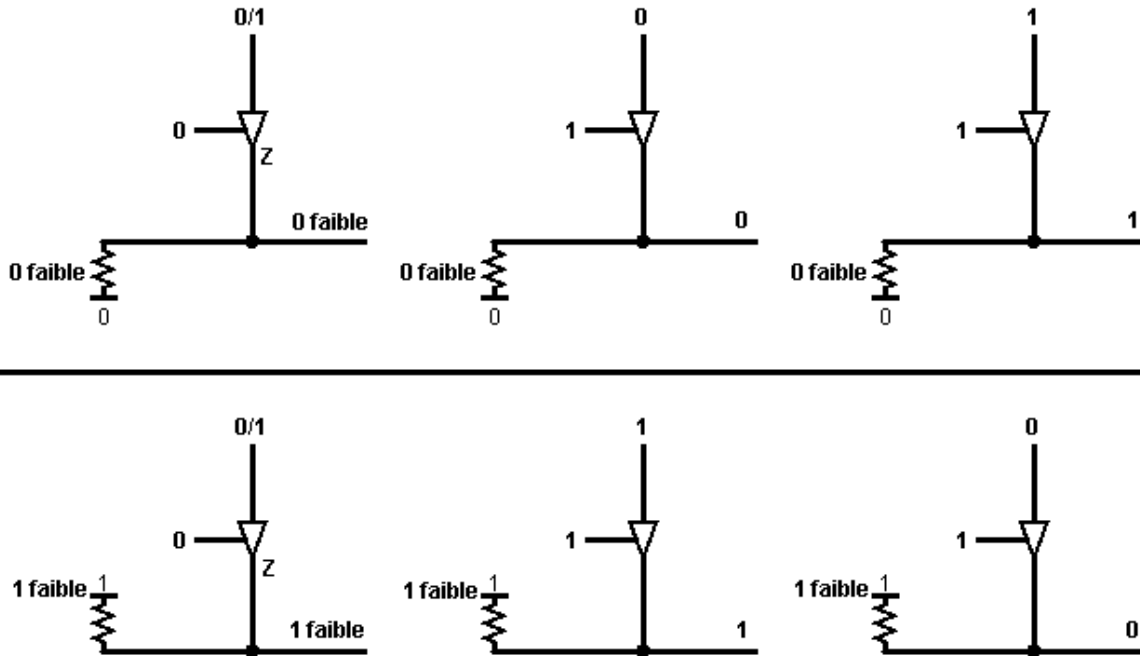
Remarque 4: L'utilisation du bus à base d'état Z est le plus souvent réservée aux systèmes simples ou aux anciens systèmes. Pour les systèmes actuels ou plus complexes, les Multiplexeurs/Démultiplexeurs sont généralement plus appropriés, ou même plus pour les systèmes les plus performants on utilise des connexions points-à-points entre paires de circuits.

Remarque 5: Il est possible de trouver l'Arbitre comme un circuit à part, comme il est possible de le trouver intégré à l'intérieur du Processeur, et parfois il est simplement implémenté par un décodeur. Dans les cartes-mères modernes, on peut dire que les 2 chipsets North-Bridge et South-Bridge possèdent des fonctions d'Arbitrage de Bus.

Les états faibles 0 et 1 :

- La valeur Flottante peut avoir des valeurs comme 1 Volt, 2,5 Volts, 3,8 Volts, et ces valeurs sont interdites dans les systèmes binaires numériques.
- Pour certains circuits, l'utilisation de Z comme valeur en entrée pose problème, et peut aboutir au dysfonctionnement du circuit.
- En plus ne pas avoir une valeur stable en entrée est une mauvaise manière de concevoir les circuits numériques.
- C'est pour toutes ces raisons que les concepteurs de Hardware utilisent un 4-ème et un 5-ème état après le 0, le 1 et le Z, qui sont le 0-*faible* et le 1-*faible*.

- Le 0-*faible* et le 1-*faible* sont des constates et ne peuvent pas changer comme les autres signaux (0,1 et Z).
- Ils sont généralement utilisés pour supplanter (remplacer) la valeur Z dans un bus vide (n'effectue pas de transmission) et offrir une entrée stable pour les circuits.



- Les 3 Bus en haut sont branchés à la valeur constante 0-*faible*, et les 3 Bus en bas sont branchés à la valeur constante 1-*faible*.
- Pour fournir une valeur faible sur un fil, il suffit de brancher une résistance à la borne 0 Volt (ou GND) pour la constante 0-*faible*, et une résistance à 5 Volt (ou Vcc) pour la constante 1-*faible*.
- Dans les deux bus à gauche le contrôleur du Tristate Buffer est à 0 et fait sortir Z, dans ce cas la valeur Z est supplantée par 0-*faible* dans le bus en haut et par 1-*faible* en bas.
- Sur les deux bus au milieu il n'y a pas conflit, puisque pour celui en haut le Tristate Buffer fait passer la valeur 0, et le bus contient 0-*faible*, et inversement en bas, le Tristate Buffer fait passer la valeur 1, et le bus contient 1-*faible*.
- Pour les deux bus à droite, il y a conflit sur les deux, en haut le Tristate Buffer fait passer la valeur 1 et le bus contient un 0-*faible*, dans ce cas c'est le 1 qui occupe le bus. Et la même chose en bas, le Tristate Buffer fait passer la valeur 0 et le bus contient un 1-*faible*, c'est le 0 qui occupe le bus.
- La règle est que si un 0-*faible* ou un 1-*faible* entre en conflit avec l'état Z, c'est 0-*faible* ou 1-*faible* qui l'emporte. Dans l'autre cas, si 0-*faible* ou 1-*faible* entre en conflit avec 1 ou 0, c'est le 1 ou 0 qui l'emporte.
- La relation de domination est : $(0\wedge 1) > (0\text{-faible}\wedge 1\text{-faible}) > Z$. (\wedge est un ou, $>$ est la direction de domination).

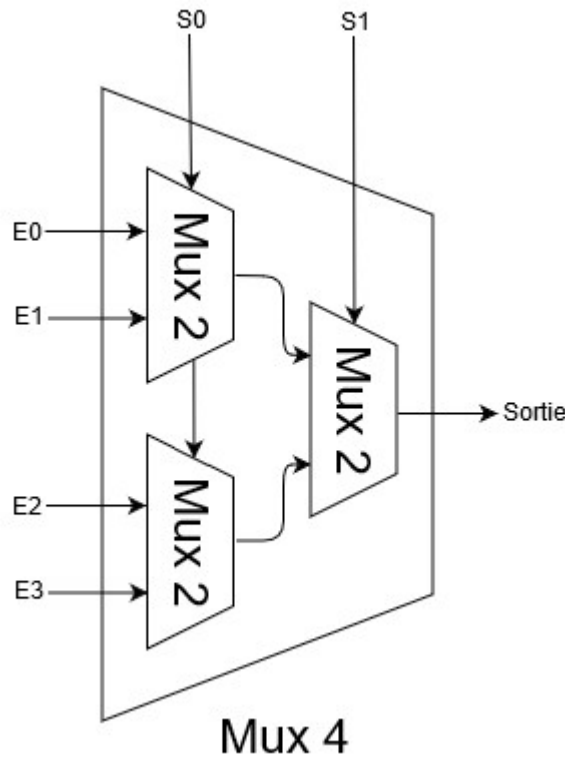
Remarque 1: La résistance branchée sur le Vcc pour fournir le 1-*faible* est appelée résistance PullUp, par convention elle doit être représentée orientée vers le haut sur les schémas. Et inversement, la résistance branchée sur le GND pour fournir le 0-*faible* est appelée résistance PullDown, et par convention sur les schémas doit être représentée en l'orientant vers le bas. Dans la pratique la résistance peut avoir une valeur entre $1k\Omega$ - 10Ω .

Remarque 2: La compréhension de pourquoi l'ajout d'une résistance produit un signal faible qui s'annule en conflit avec un signal normal entre dans le domaine de l'électronique analogique, qui ne sera pas expliqué ici. Malgré ça, son fonctionnement reste relativement simple et il est possible pour un étudiant avec ses connaissances de la deviner lui-même.

11. La conception Hardware par la composition en cascade

- Toutes les technologies complexes utilisent dans leurs conceptions ce qu'on appelle la construction en composition, ou la modularité.
- Le concept a été décrit dans la première règle de la construction d'un circuit combinatoire, c'est que le circuit combinatoire se compose de plusieurs circuits combinatoires de petite taille, eux-mêmes se composent d'autres circuits encore plus petits et ainsi de suite jusqu'aux portes logiques.
- C'est le même concept de fragmenter un gros programme en plusieurs fonctions dans la conception Software.
- Mais la composition en cascade (ou en Slice) consiste à faire de la construction d'un circuit avec des petits composants du même type de ce circuit.
- Par exemple la construction d'un Multiplexeur de 4 entrées se fait avec des Multiplexeurs de petite taille de 2 entrées.
- Cette manière de conception est très pratique et très flexible pour la construction facile des circuits avec des tailles variées. Malheureusement cette technique n'est applicable que pour certains circuits spécifiques et non tous les circuits.

Exemple 1: La construction de Multiplexeur 4 à partir de Multiplexeurs 2.



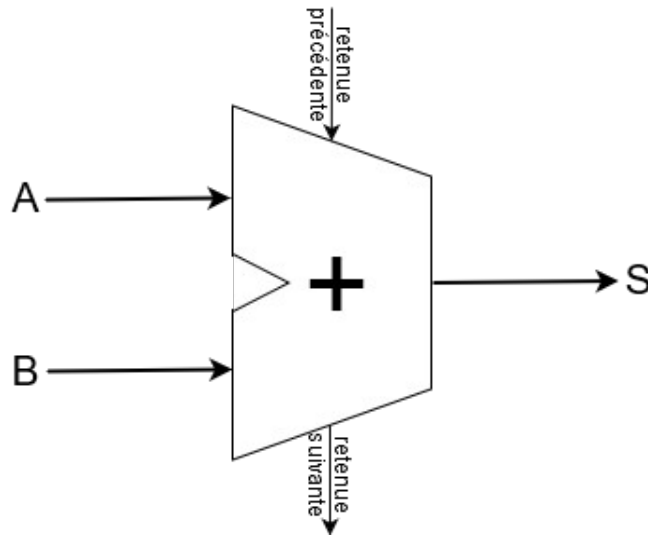
Exemple 2: La construction d'un additionneur 4 bits Complément-à-2/Valeur Absolue à partir de d'additionneur 1 bit.

$$\begin{array}{r}
 \begin{array}{cccc}
 & \overset{1}{\leftarrow} & \overset{1}{\leftarrow} & \\
 0 & 0 & 1 & 1 \\
 + & 0 & 0 & 1 & 1 \\
 \hline
 = & 0 & 1 & 1 & 0
 \end{array}
 \end{array}$$

Si on observe l'opération d'addition Complément-à-2/Valeur Absolue sur un seul bit, par exemple sur le 2-ème bit (2-ème colonne sur le schéma), on peut construire un additionneur sur 1 bit qui doit avoir 3 entrées qui sont le 2-ème chiffre dans A, le 2-ème chiffre dans B et la retenue du calcul du bit précédent (du 1-ier bit). Et 2 sorties qui sont la Somme et la retenue pour le calcul du bit suivant (pour le 3-ème bit).

Remarque: L'additionneur pour le Complément-à-2 et l'additionneur pour la Valeur Absolue sont le même additionneur, qui peut faire l'addition sur les deux types d'encodage, par contre les additionneurs pour Signe et Valeur Absolue et Complément-à-1 sont construits différemment.

Étape 1 : Schéma global



Étape 2 : Table de Vérité

A	B	R _p	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

R_p : Retenue précédente

R_s : Retenue suivante

Étape 3 : Fonctions Canoniques Disjonctives

$$S(A,B,R_p) = \bar{A} \cdot \bar{B} \cdot R_p + \bar{A} \cdot B \cdot \bar{R}_p + A \cdot \bar{B} \cdot \bar{R}_p + A \cdot B \cdot R_p$$

$$R_s(A,B,R_p) = \bar{A} \cdot B \cdot R_p + A \cdot \bar{B} \cdot R_p + A \cdot B \cdot \bar{R}_p + A \cdot B \cdot R_p$$

Étape 4 : Minimisation Algébrique

$$S(A,B,R_p) = \bar{A} \cdot \bar{B} \cdot R_p + \bar{A} \cdot B \cdot \bar{R}_p + A \cdot \bar{B} \cdot \bar{R}_p + A \cdot B \cdot R_p$$

$$S(A,B,R_p) = \bar{A} \cdot (\bar{B} \cdot R_p + B \cdot \bar{R}_p) + A \cdot (\bar{B} \cdot \bar{R}_p + B \cdot R_p)$$

$$S(A,B,R_p) = \bar{A} \cdot (B \oplus R_p) + A \cdot (B \otimes R_p)$$

$$S(A,B,R_p) = \bar{A} \cdot (B \oplus R_p) + A \cdot (B \oplus R_p)$$

$$S(A,B,R_p) = A \oplus B \oplus R_p$$

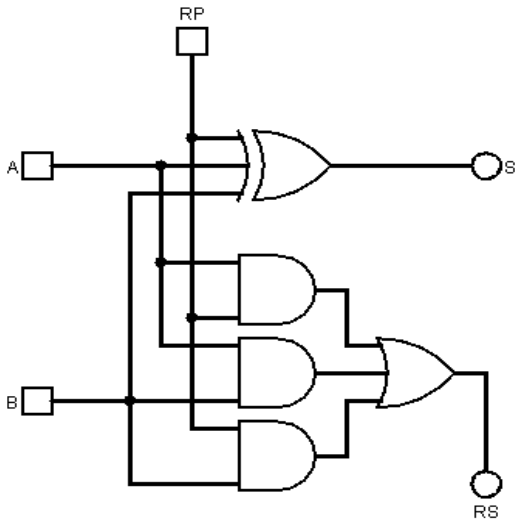
$$R_s(A,B,R_p) = \bar{A} \cdot B \cdot R_p + A \cdot \bar{B} \cdot R_p + A \cdot B \cdot \bar{R}_p + A \cdot B \cdot R_p$$

$$R_s(A,B,R_p) = \bar{A} \cdot B \cdot R_p + A \cdot \bar{B} \cdot R_p + A \cdot B \cdot \bar{R}_p + A \cdot B \cdot R_p + A \cdot B \cdot R_p + A \cdot B \cdot R_p$$

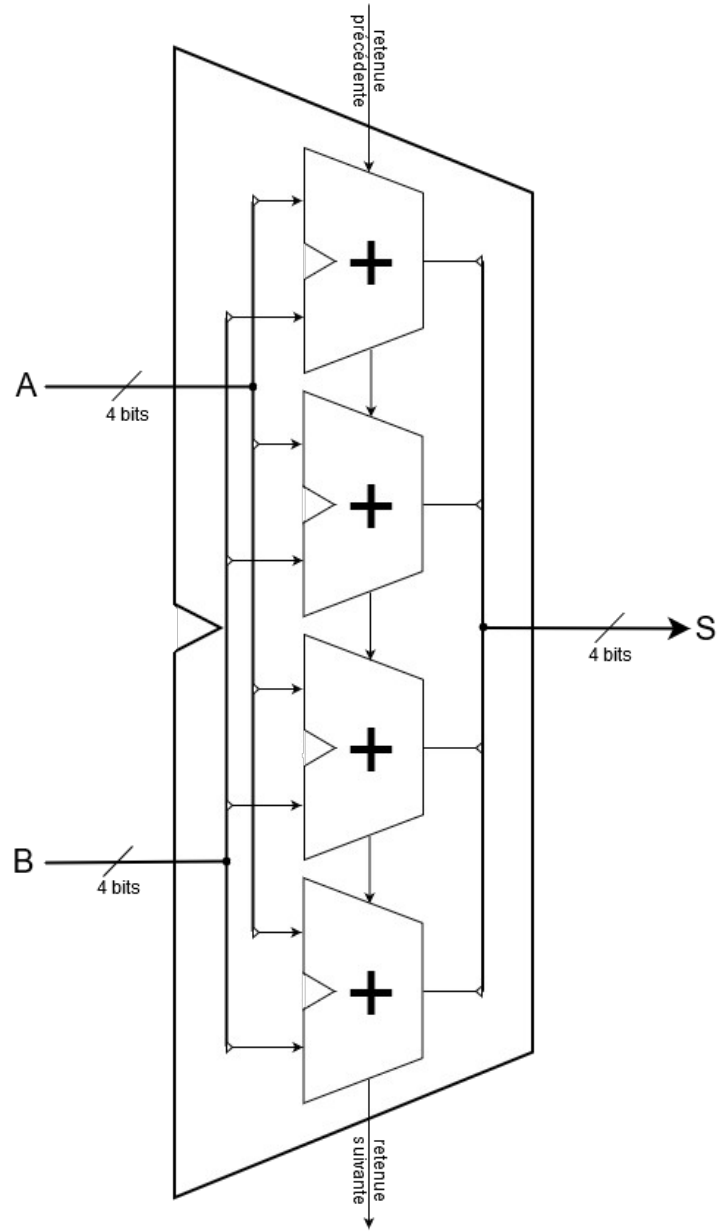
$$R_s(A,B,R_p) = (\bar{A} \cdot B \cdot R_p + A \cdot \bar{B} \cdot R_p) + (A \cdot \bar{B} \cdot R_p + A \cdot B \cdot R_p) + (A \cdot B \cdot \bar{R}_p + A \cdot B \cdot R_p)$$

$$R_s(A,B,R_p) = (B \cdot R_p) + (A \cdot R_p) + (A \cdot B)$$

Étape 5 : Logigramme



Logigramme de l'additionneur 4 bits en Cascade



Sur le schéma de l'additionneur en cascade, chaque additionneur 1 bit représente une colonne (1 bit) sur l'opération de l'addition sur le schéma précédent. On peut aussi observer sur le schéma actuel comment la retenue suivante d'un additionneur en sortie est passée comme entrée en retenue précédente pour le bit suivant.

Remarque 1: Le petit triangle sur les bus à 4 bits A, B et S sont des indicateurs pour dénoter qu'un seul fil parmi les 4 fils a été extirpé du bus.

Remarque 2: L'additionneur sur 4 bits représenté ici est différent de celui présenté dans la section 8.5 sur l'entrée retenue précédente que celui de la section 8.5 n'a pas. L'avantage de l'additionneur présenté ici est qu'il adhère aux méthodes de la conception en cascade, ainsi on peut raccorder en cascade 2 additionneurs 4 bits pour avoir un additionneur de 8 bits et ainsi de suite pour avoir l'additionneur sur 12 bits, 16 bits...etc.

Remarque 3: La condition pour que l'additionneur sur 4 bits fonctionne correctement est que l'entrée retenue précédente du premier additionneur doit être mise à 0, le premier bit n'a pas de retenue précédente.

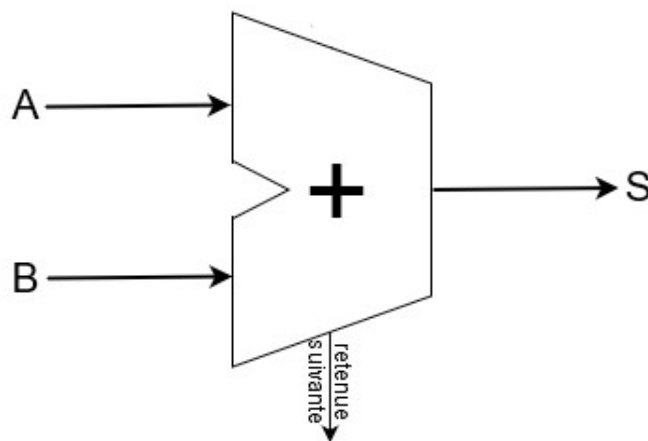
Remarque 4: La principale raison du succès de l'encodage Complément-à-2 par rapport aux autres encodages signés, est qu'avec l'encodage Complément-à-2 le Hardware n'a pas besoin de soustracteur, la soustraction est effectuée par l'addition de A avec (-B), contrairement à l'encodage Signe et Valeur Absolue qui exige un soustracteur dédié. Pour le Complément-à-1, le soustracteur peut aussi être remplacé par un additionneur, mais il existe des situations où il doit faire l'addition en deux fois.

Remarque 5: L'additionneur sur 1 bit qu'on vient de voir est appelé *Full-Adder* (Additionneur Complet), il existe aussi un autre type d'additionneur sur 1 bit, appelé *Half-Adder* (Demi-Additionneur), il sera présenté dans l'exemple suivant.

Exemple : Demi-Additionneur

Un demi-additionneur est un additionneur sur 1 bit qui n'a pas d'entrée retenue précédente, il fait l'addition sans retenue précédente.

Étape 1 : Schéma global



Étape 2 : Table de Vérité

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

R_s : Retenue suivante

Étape 3 : Fonctions Canoniques Disjonctives

$$S(A,B) = \bar{A} \cdot B + A \cdot \bar{B}$$

$$R_s(A,B) = A \cdot B$$

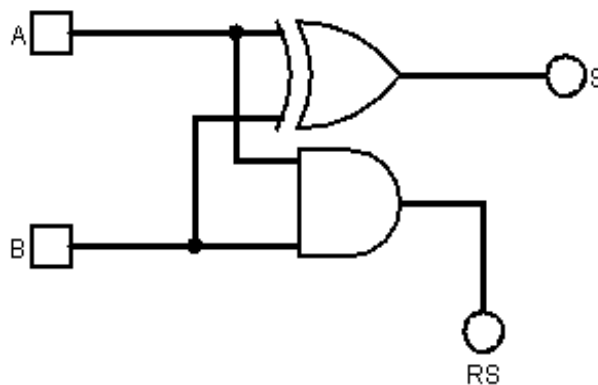
Étape 4 : Minimisation Algébrique

$$S(A,B) = \bar{A} \cdot B + A \cdot \bar{B}$$

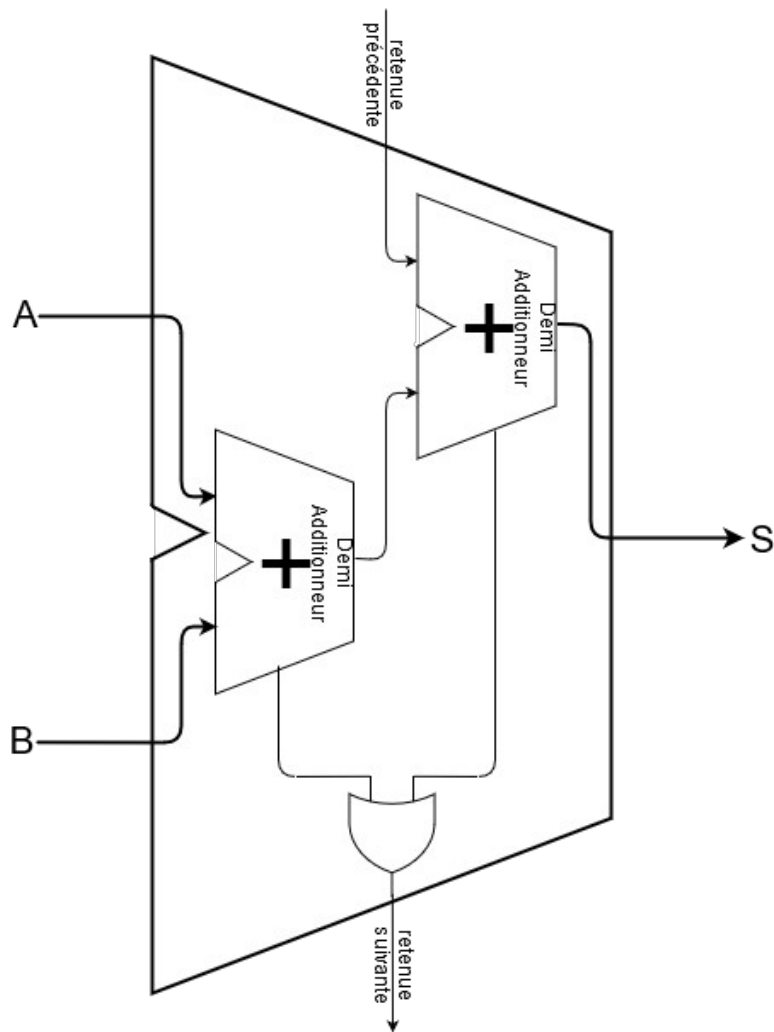
$$S(A,B) = A \oplus B$$

$$R_s(A,B) = A \cdot B$$

Étape 5 : Logigramme



Il est possible de construire un Full-Adder à partir de 2 Half-Adder comme sur le schéma suivant, d'où le nom Demi-Additionneur.



Additionneur Complé

Remarque 1: Il est aussi possible de construire des Additionneurs à plusieurs bits avec des Demi-Additionneurs.

Remarque 2: Le terme *Pin* désigne une entrée ou une sortie numérique, son origine vient du domaine de l'électronique où physiquement ça représente la patte d'un circuit intégré soudée sur un circuit imprimé.