Réalisation de Circuits logiques

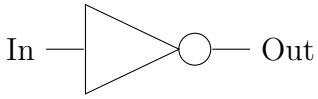
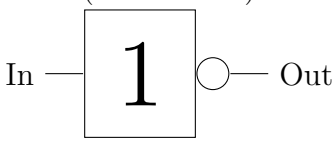
Cette fiche d'activité a pour but de répondre à la question suivante : Comment la machine manipule-t-elle le langage binaire pour faire de la logique et du calcul ?
La réponse repose essentiellement sur un composant électronique très particulier, le **transistor**.

1. Portes logiques

En considérant que le 1 correspond à une présence de courant électrique et que le 0 correspond à une absence de courant, on peut, à l'aide des transistors, transformer un 1 en 0, et un 0 en 1.
Ainsi on réalise la fonction logique NON, dont la table de vérité est la suivante :

In	Out
0	1
1	0

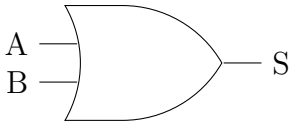
Le circuit qui réalise la fonction logique NON est appelé inverseur. On parle de **porte logique**, et la porte **NON** est représentée de la manière suivante dans un **circuit logique** :

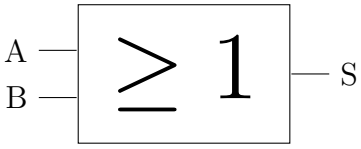
	Représentation distinctive (norme ANSI/IEEE)	Représentation rectangulaire (norme CEI)	Table de vérité
NON			

In	Out
0	1
1	0

Le cercle, commun aux deux représentations, symbolise l'inversion du signal.
On peut réaliser d'autres portes logiques :

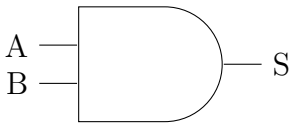
OU (OR)

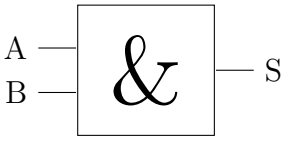




A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

ET (AND)

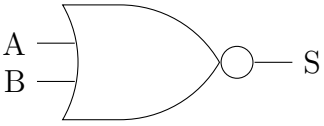


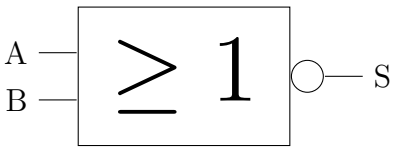


A	B	S
0	0	0
0	1	0
1	0	0
1	1	1

À partir de ces trois portes (en fait, même à partir de la porte NON et de l'une des deux portes OU et ET), on peut construire d'autres portes logiques :

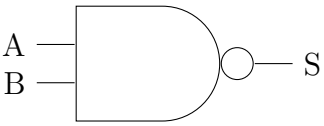
NOR (non OU)

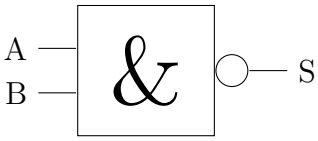




A	B	S
0	0	1
0	1	0
1	0	0
1	1	0

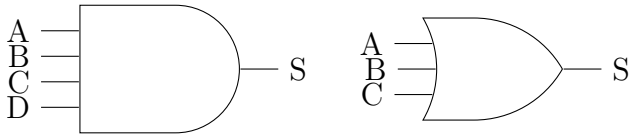
NAND (non ET)





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

Note : on peut considérer des portes avec plusieurs entrées, comme ici les portes ET et OU :



2. Fonctions logiques et opérations

On utilise certaines notations pour les opérations logiques (booléennes) :

- A ET B se note AB (comme un produit)
- A OU B se note $A + B$ (comme une somme)
- NON A se note $\overline{A}$ (comme un événement contraire)

Ainsi, par exemple « A NOR B » peut se noter $\overline{A + B}$.
Étant donnée une table de vérité d'une fonction logique, on peut en déduire une formule logique qui lui correspond. Pour cela, il suffit de considérer les conditions pour lesquelles la sortie est 1. Chaque condition se traduit par un ET, et on ajoute (avec des OU) les différentes conditions.

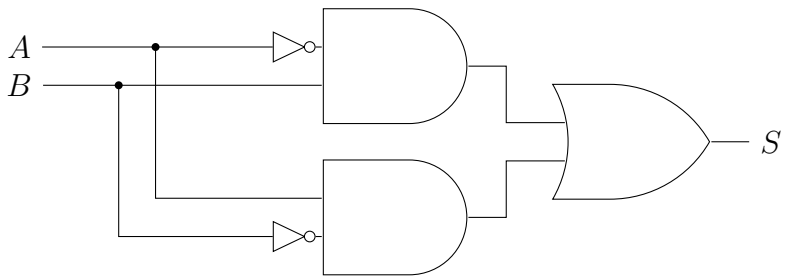
Exemple Prenons le cas de la fonction XOR (le OU exclusif, qui ne vaut 1 que lorsqu'une seule des deux entrées vaut 1), dont la table de vérité est la suivante :

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

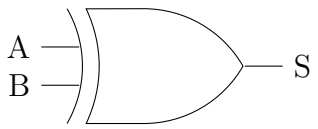
Une formule logique qui correspond est alors $\overline{A}B + A\overline{B}$.

Une fois une telle formule à disposition, on peut alors faire le circuit logique correspondant : Il suffit d'utiliser des portes ET avec éventuellement des négations avant, et enfin rassembler toutes les sorties dans les entrées d'une porte OU.

Exemple Un circuit logique réalisant le XOR vu à l'exemple précédent est donc :



Le symbole de ce circuit sous forme de porte logique est le suivant :



Exercice 1
Suivre la méthode développée ci-dessus pour donner une formule logique puis un circuit logique correspondant à la fonction XNOR dont la table de vérité est la suivante :

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

Exercice 2
En utilisant la même méthode, déterminer un circuit réalisant la fonction NAND.

Exercice 3 (Facultatif)
Chercher un autre circuit réalisant la fonction XNOR.

3. Application : Additionneur 1 bit

Dans le processeur d'un ordinateur, se trouve une unité arithmétique et logique (UAL), dont le but est de réaliser les calculs arithmétiques et logiques.
Elle contient donc des circuits logiques comme ceux évoqués plus haut, et d'autres plus complexes.
En tant qu'unité arithmétique, elle est capable d'effectuer la somme de deux nombres (en binaire).

a. demi-additionneur

Le demi-additionneur est une fonction qui prend deux bits en entrée et qui en sortie renvoie le résultat de leur somme en ignorant l'éventuelle retenue.

Exercice 4

1. Reproduire et compléter la table de vérité ci-contre du demi-additionneur.
2. De quelle fonction logique, déjà étudiée précédemment, cette table est-elle la représentation ?
   Nous connaissons donc déjà un circuit réalisant cette fonction.

A	B	S
0	0	
0	1	
1	0	
1	1	

b. additionneur complet 1 bit

Un additionneur (complet) 1 bit est une fonction qui prend trois bits (A,B et Re) en entrée, et qui en sortie renvoie deux bits (S et Rs). Il fait la somme des entrées A et B en prenant en compte une retenue Re, et renvoie la somme S obtenue avec la valeur de la retenue (Rs).

Exemples

- Pour $A = 1$, $B = 0$ et $Re = 1$, on obtient $S = 0$ et $Rs = 1$;
- Pour $A = 1$, $B = 1$ et $Re = 1$, on obtient $S = 1$ et $Rs = 1$;

Exercice 5

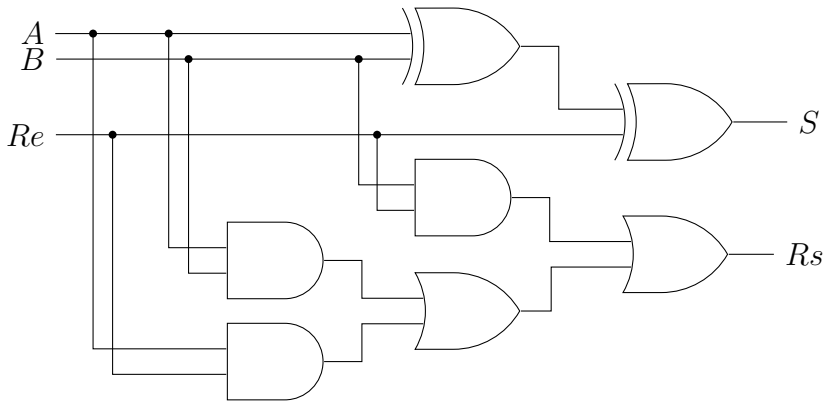
1. Établir la table de vérité de l'additionneur complet 1 bit (elle contient 8 lignes).

A	B	Re	S	Rs
0	0	0	...	...
...	...	...	...	...

2. (a) Établir une formule logique donnant S en fonction de A, B et Re.
   (b) Établir alors un circuit logique.
3. (a) Établir une formule logique donnant Rs en fonction de A, B et Re.
   (b) Compléter le circuit précédent pour ajouter la sortie Rs.

I. Application : Additionneur 4 bits

Le but de cette section est de réaliser le circuit logique correspondant à l'additionneur 4 bits en utilisant le logiciel Logisim (http://www.cburch.com/logisim/)
Par des calculs (booléens), il est possible de simplifier les formules logiques, et donc les circuits qui leur correspondent, surtout si l'on se permet d'utiliser d'autres portes logiques que le NON, le ET et le OU.
En particulier, voici une forme un peu plus simple du circuit logique correspondant à l'additionneur 1 bit que celle obtenue à la section précédente :



Exercice 6
Réaliser les étapes suivantes en utilisant le logiciel Logisim.

1. Réaliser le circuit donné ci-dessus.
   Les bits d'entrée et de sortie sont disponibles directement dans la barre d'outils horizontale située sous le menu.
   Les portes logiques sont situées dans la fenêtre à gauche, dans le dossier « Gates ».
   Pour créer les fils, il suffit de cliquer sur une entrée/sortie et de tirer jusqu'à destination si un coude suffit, sinon on les fait en deux temps.
   En cas de réelle difficulté, le circuit pourra être donné.
2. Vérifier que la table de vérité du circuit correspond bien à celle obtenue à la section précédente.
   Pour cela, aller dans le menu Project → Analyze Circuit.
3. Nous allons utiliser le circuit obtenu comme un composant d'un circuit plus complexe.
   Il est possible de configurer son apparence lorsqu'on l'utilise comme composant d'un autre circuit.
   Pour cela, cliquer sur le dernier élément de la deuxième ligne de la barre d'outils (Edit viewed circuit's subcircuit appearance). En particulier bien placer les entrées et sorties pour savoir où sont chacun des éléments.
4. Ajouter un circuit (menu Project → Add circuit...) appelé Add4bits
   Utiliser le circuit précédent pour créer un additionneur 4 bits.
   Cet additionneur prend deux ensembles de 4 bits à sommer et un bit de retenue, et renvoie un ensemble de 4 bits et un bit de retenue.
5. (facultatif) Pour aller un peu plus loin, il est possible de rassembler ou de séparer des bits avec l'élément « Splitter » du dossier Wiring.
   On peut alors utiliser des entrées et sorties sur 4 bits, ce qui permet de voir les valeurs d'entrées et sorties en nombres binaires. Cependant ainsi on perd la possibilité d'avoir les tables de vérité.