Comme on l'a déjà vu, toute donnée informatique est encodée dans une machine par une succession de 0 et de 1. Lorsqu'une machine effectue une opération, elle part d'une certaine configuration de sa mémoire et termine dans une nouvelle configuration.

Cela nous pousse à étudier une certaine classe d'objets mathématiques décrivant de telles transformation : les fonctions booléennes. Après avoir décrit ces objets, nous verrons que ces objets mathématiques peuvent être physiquement réalisés par des circuits électriques dits logiques, circuits étant au fondement de l'architecture des ordinateurs.

1 Fonctions booléennes

1.1 Tables de vérité

De manière générale, un booléen est un système pouvant se trouver dans deux états. C'est le système le plus simple pouvant contenir de l'information. On représente en général ces deux états par 0 et 1, les booléens peuvent donc être vus comme l'ensemble

$$\mathbb{B} = \{0, 1\}$$

Sur l'ensemble $\mathbb{B}$, il est possible de définir, comme sur tout autre ensemble, des fonctions appelées ici **fonctions booléennes**. Une fonction booléenne f est un objet associant à chaque booléen ou $n-$uplet de booléens un unique booléen image

$$f : \mathbb{B}^n \longrightarrow \mathbb{B}$$

Pour définir une telle fonction, une possibilité est de donner la liste des images pour chaque $n-$uplet de booléens. On appelle une telle liste la **table de vérité** de f. Voici un exemple de fonction booléenne décrite par sa table de vérité :

x	y	$f(x,y)$
0	0	0
0	1	0
1	0	1
1	1	0

On lit sur la table que f est une fonction à deux arguments (x et y), on trouve donc dans sa table dans les deux premières colonnes $2^2 = 4$ configurations possibles pour ceux-ci. La troisième colonne donne quant à elle les images associées par f à ces configurations. On y lit par exemple à la troisième ligne que

$$f(1, 0) = 1$$

et à la quatrième ligne que

$$f(1, 1) = 0$$

Remarque : L'ordre dans lequel ont été placées les configurations n'est pas aléatoire. On peut en effet interpréter la donnée de n arguments booléens comme un nombre entier compris entre 0 (0...0) et $2^n - 1$ (1...1), on donne donc les configurations en respectant cette numérotation.

1.2 Fonctions et, ou, non

On peut interpréter les booléens comme des valeurs de vérité (0 = Faux, 1 = Vrai) et donner à certaines fonctions simples un nom faisant référence à cette interprétation logique.

On définit ainsi les fonctions **non**, **et** et **ou** :

x	non(x)
0	1
1	0

La fonction **non** renvoie la valeur inverse de son argument : 0 devient 1 et inversement. Cela correspond à la **négation** logique (ne pas...). Si x correspond à la phrase « il pleut », non(x) correspond à « il ne pleut pas ». On notera aussi

$$non(x) = \overline{x}$$

x	y	et(x,y)	ou(x,y)
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

La fonction **et** renvoit 1 dès lors que ses deux arguments valent 1 et 0 sinon. Cela correspond à la **conjonction** logique. Si x correspond à la phrase « il pleut » et y à « nous sommes mardi », et(x, y) correspond à « il pleut et nous sommes mardi ». On notera aussi

$$et(x, y) = x \wedge y$$

La fonction **ou** renvoit 1 dès lors que l'un au moins de ses deux argument vaut 1. Cela correspond à la **disjonction** logique. Si x correspond à la phrase « il pleut » et y à « nous sommes mardi », ou(x, y) correspond à « il pleut ou nous sommes mardi ». On notera aussi

$$ou(x, y) = x \vee y$$

1.3 Expressions booléennes

Décrire une fonction booléenne par sa table de vérité n'est pas très pratique, en effet plus le nombre de variables augmente, plus le nombre de configurations devient grand (2^n configurations pour n variables). Aussi on utilise plus volontiers des **expressions booléennes**.

Une expression booléenne est une succession de booléens (0 ou 1), de variables booléennes (x, y, z, a, ...) liées entre elles par les symboles $\wedge$, $\vee$, $\overline{\ }$ et des parenthèses. Par exemple

$$(x \wedge y) \vee \overline{z}$$

On peut **évaluer** une expression booléenne. Cela correspond à **assigner** à chacune des variables impliquées une valeur (0 ou 1) et à mener le calcul jusqu'à trouver un **résultat** en se servant des tables de vérité de et ou et non.

Par exemple, évaluons l'expression précédente pour $x = 1$, $y = 0$ et $z = 0$ (valeurs choisies au hasard)

$$(x \wedge y) \vee \overline{z} = (1 \wedge 0) \vee \overline{0} = 0 \vee \overline{0} = 0 \vee 1 = 1$$

Il devient alors possible de définir des fonctions booléennes en donnant non plus une table de vérité mais en donnant une expression booléenne. Par exemple, on peut définir la fonction f de par partie 1.1 par

$$f(x, y) = x \wedge \overline{y}$$

En particulier, l'évaluation de l'expression de f pour chaque affectation possible de (x, y) en fournit la table de vérité.

On dit que deux expressions booléennes (resp. fonctions) sont **égales** si elles amènent au même résultat quelque soit l'évaluation (resp. si elles ont la même table de vérité). Dès lors que deux expressions sont égales, il est possible de remplacer l'une par l'autre dans n'importe quelle partie d'une expression booléenne où on les identifie. Voyons les quelques égalités formant les règles de base du **calcul booléen**.

involution :	$\overline{\overline{x}} = 0$	
neutre :	$x \wedge 1 = x$	$x \vee 0 = x$
absorbant :	$x \wedge 0$	$x \vee 1 = 1$
idempotence :	$x \wedge x = x$	$x \vee x = x$
complément :	$x \wedge \overline{x} = 0$	$x \vee \overline{x} = 1$
commutativité :	$x \wedge y = y \wedge x$	$x \vee y = y \vee x$
associativité :	$x \wedge (y \wedge z) = (x \wedge y) \wedge z$	$x \vee (y \vee z) = (x \vee y) \vee z$
distributivité :	$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$	$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$
De Morgan :	$\overline{x \wedge y} = \overline{x} \vee \overline{y}$	$\overline{x \vee y} = \overline{x} \wedge \overline{y}$

Pour démontrer de telles égalités, on évalue les expression des deux côtés du signe $=$ sur toutes les configurations possibles. Cela revient à dresser les tables de vérité des fonctions associées afin de constater que ce sont les mêmes. Démontrons par exemple la loi de De Morgan pour $\wedge$ (on décompose les calculs en ajoutant des colonnes intermédiaires dans la table) :

x	y	$x \wedge y$	$\overline{x \wedge y}$	$\overline{x}$	$\overline{y}$	$\overline{x} \vee \overline{y}$
0	0	0	**1**	1	1	**1**
0	1	0	**1**	1	0	**1**
1	0	0	**1**	0	1	**1**
1	1	1	**0**	0	0	**0**

Il devient possible, une fois ces égalités démontrées, de démontrer de nouvelles égalités en les utilisant. Démontrons par exemple l'égalité

$$(a \vee b) \wedge (a \vee \overline{b}) = a$$

Par **distributivité** du $\vee$ sur $\wedge$ on peut factoriser l'expression par a à gauche, soit

$$(a \vee b) \wedge (a \vee \overline{b}) = a \vee (b \wedge \overline{b})$$

Puis, par **complémentaire** sur $\wedge$, on a $b \wedge \overline{b} = 0$, d'où

$$(a \vee b) \wedge (a \vee \overline{b}) = a \vee 0$$

Enfin, par **neutralité** de 0 pour $\vee$ on trouve l'égalité voulue

$$(a \vee b) \wedge (a \vee \overline{b}) = a$$

Pour démontrer une égalité entre deux expressions ou fonctions booléennes, on dispose de deux méthodes :

1. On dresse puis on compare les tables de vérités.
2. On utilise les règles du calcul booléen pour passer d'une expression à l'autre.

1.4 Théorème fondamental

Le théorème suivant affirme l'universalité, au sein des fonctions booléennes des opérateurs $\wedge$, $\vee$ et $\overline{\ }$.

Théorème : Toute fonction booléenne peut s'exprimer par une expression booléenne (ne comportant que $\wedge$, $\vee$ et $\overline{\ }$).

Le preuve de ce théorème ne sera pas détaillée ici. Elle repose sur un raisonnement par récurrence (*cf* cours de terminale maths) ainsi que sur l'utilisation de la fonction booléenne multiplexeur définie par

$$mux(a, b, s) = (s \wedge a) \vee (\overline{s} \wedge b)$$

Comme expliqué en introduction, toute opération informatique peut être représentée par des fonctions booléennes. Le théorème fondamental permet d'aller plus loin : toute opération informatique peut être représentée par une combinaison plus ou moins complexe d'expressions booléennes à l'aide des opérateurs $\wedge$, $\vee$ et $\overline{\ }$.

2 Circuits logiques

Les circuits logiques sont des circuits électriques permettant de réaliser physiquement des fonctions booléennes. Ils sont composés d'**entrées** correspondant aux arguments des fonctions et de **sorties** correspondant à leurs résultats. On les conçoit en utilisant des **portes logiques**, elles même conçues en utilisant un composant de base très simple : le **transistor**.

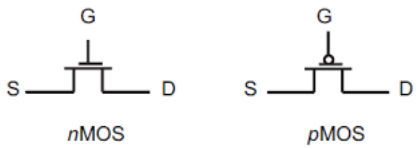
2.1 Transistors et circuits logiques

Un **transistor** est un composant électrique à trois branches appelées grille (G) source (S) et drain (D) agissant comme un interrupteur : en fonction de la tension présente ou non sur la grille, le courant passera ou non de la source vers le drain.
Il en existe deux types, les nMOS et les pMOS :

— Les transistors nMOS laissent passer le courant de S vers D lorsqu'un certaine tension V_s est appliquée en G et les isole sinon.

— Les transistors pMOS laissent passer le courant de S vers D lorsqu'aucune tension V_s n'est appliquée en G et les isole sinon.

Dans les schémas, les symboles de ces deux types de transistors sont les suivants :



Les **circuits logiques** sont des circuits électriques disposant :

— d'une source de tension V_s notée et interprétée comme le booléen 1 ;

— d'une masse de tension 0 notée et interprétée comme le booléen 0 ;

— d'une ou plusieurs **entrées** pouvant présenter des tensions de V_s ou 0 ;

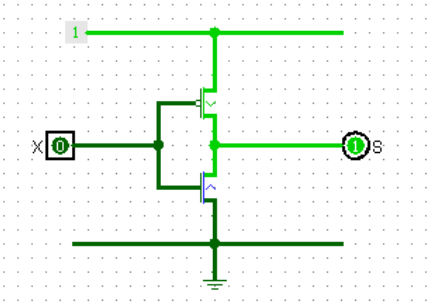
— d'une ou plusieurs **sorties** dont la tension dépend des entrées ;

— d'un certain nombre de transistors nMOS et pMOS.

L'état des différentes sorties (0 ou 1) dépendant des états des entrées (0 ou 1), les circuits logiques réalisent de manière physique les fonctions booléennes. Afin de réaliser n'importe lequel de ces circuits nous allons nous aider dans un premier temps des transistors afin de réaliser des sous-circuits de base correspondant aux fonction booléennes universelle ET OU et NON : les **portes logiques**.

2.2 Portes logiques

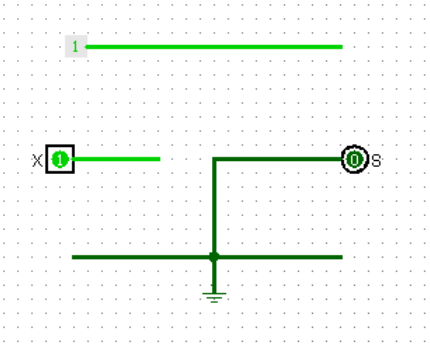
Porte NON Notre premier objectif est de concevoir un circuit logique ayant le comportement de l'opérateur non. Le voici :

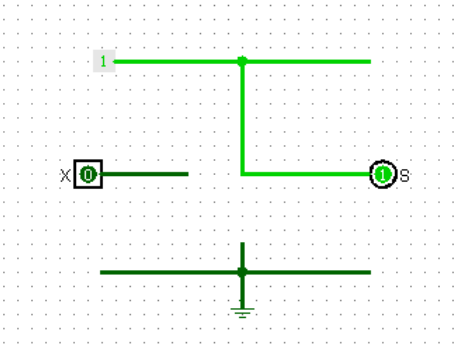


Vérifions que la sortie S de ce circuit vaut toujours $\overline{X}$:

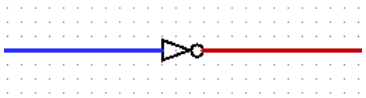
— Si $X = 1$ alors étant donné le comportement des transistors, S est connecté à 0 et pas à 1 : $S = 0$.

— À l'inverse, si $X = 0$ alors étant donné le comportement des transistors, S est connecté à 1 et pas à 0 : $S = 1$.

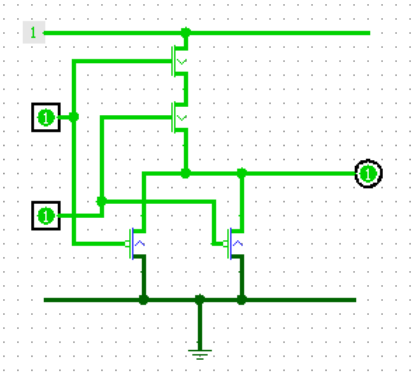




On peut maintenant utiliser dans nos circuits la **porte NON** dont le symbole est



Porte ET Le circuit logique reproduisant le comportement de la fonction ET est le suivant :

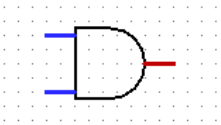


Vérifions que la sortie S de ce circuit vaut toujours $X \wedge Y$:

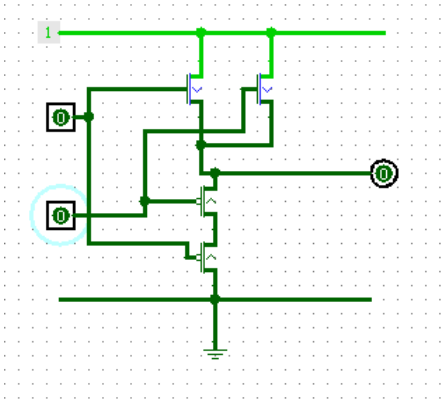
— Si X et Y valent simultanément 1, alors les deux nMOS en série sont fermés et connectent S au 1 alors que les deux pMOS en dérivation sont ouverts et isolent donc cette dernière de la masse, d'où $S = 1$.

— Si à l'inverse l'une des deux entrées vaut 0, l'un des deux nMOS en série est ouvert et la sortie est isolée du 1 alors que l'un des deux pMOS en dérivation connecte celle-ci à la masse donc $S = 0$.

On peut maintenant utiliser dans nos circuits la **porte ET** dont le symbole est



Porte OU Le circuit logique reproduisant le comportement de la fonction OU est le suivant :

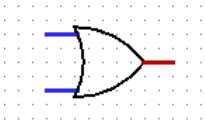


Vérifions que la sortie S de ce circuit vaut toujours $X \vee Y$:

— Si X ou Y valent 1, alors au moins l'un des deux nMOS en dérivation est fermé et connecte S au 1 alors que l'un des deux pMOS en série est ouvert et isole donc cette dernière de la masse, d'où $S = 1$.

— Si à l'inverse les deux entrées vallent 0, les deux nMOS en dérivation sont ouverts et la sortie est isolée du 1 alors que les deux pMOS fermés et en série connectent celle-ci à la masse donc $S = 0$.

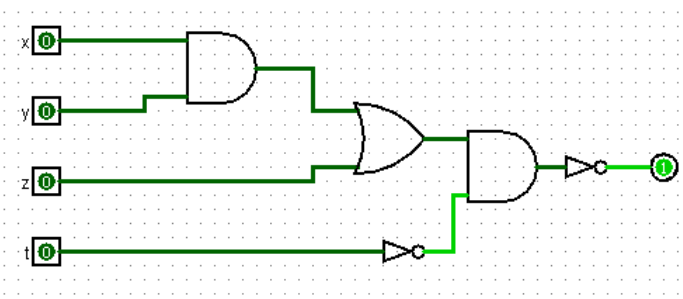
On peut maintenant utiliser dans nos circuits la **porte OU** dont le symbole est



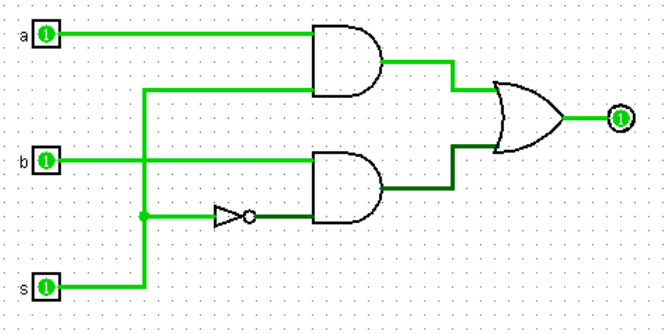
2.3 Correspondance entre expressions booléennes et circuits logiques

On donne ici des exemples de circuits logiques correspondant à des expressions booléennes.

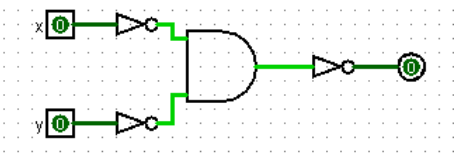
Le circuit logique suivant réalise l'expression booléenne $\overline{((x \wedge y) \vee z) \wedge \overline{t}}$:



Le circuit logique suivant réalise l'expression booléenne $mux(a, b, s) = (s \wedge a) \vee (\overline{s} \wedge b)$:



Le circuit logique suivant réalise l'expression booléenne $x \vee y = \overline{\overline{x} \wedge \overline{y}}$:



Le circuit logique suivant réalise l'expression booléenne $x \oplus y = (x \wedge \overline{y}) \vee (\overline{x} \wedge y)$: