

Portes logiques

Prérequis :

- Binaire, octet, bits.
- Connaissance des composants d'un ordinateur.
- Savoir : comprendre un algorithme.

Objectifs pour les élèves :

- Savoir : Connaître les opérations logiques de base : NOT, AND, OR, XOR. (On n'utilisera pas ici le formalisme \neg, \wedge, \vee).
- Savoir : circuits combinatoires. On utilisera ici les portes logiques sans entrer dans le détail électronique¹.
- Conformément au programme, les exercices proposés se feront sur papier sans utiliser d'ordinateur.

Pour interagir avec des ordinateurs, il faut tout d'abord un système logique assez performant, mais aussi une méthode pour communiquer avec la machine. Comment faire passer des instructions complexes à un ordinateur qui ne comprend que « 0 » et « 1 » ?

1 Opérations logiques

On se donne deux symboles distincts (classiquement 0 et 1) qu'on appelle booléens. Couramment on interprétera le booléen 0 par « faux » et le booléen 1 par « vrai ».

- L'opérateur **NOT**: **NOT** 1 = 0 et **NOT** 0 = 1.
- L'opérateur **AND**: x **AND** y renvoie 1 si x et y sont vrais, 0 sinon.
- L'opérateur **OR**: x **OR** y renvoie 1 si x ou y sont vrais (c'est-à-dire si au moins un parmi x et y est vrai), 0 sinon.
- L'opérateur **XOR**: le *ou exclusif*. x **XOR** y renvoie 1 si x ou y est vrai, mais pas les deux en même temps.

Exercice 1 : Compléter la *table de vérité* suivante :

x	y	x AND y	x OR y	NOT x	NOT y	x XOR y	(NOT x) OR (NOT y)	NOT((NOT x) OR (NOT y))
0	0							
0	1							
1	0							
1	1							

Par quelle *combinaison* des opérateurs **NOT** et **OR** peut-on remplacer l'opérateur **AND** ² ?

x	y	x TRUC y
0	0	1
0	1	0
1	0	1
1	1	1

Exercice 2 : On donne la table de vérité d'un opérateur TRUC :

Donner une expression logique pouvant correspondre à cette table de vérité.

x	y	x MACHIN y
0	0	1
0	1	0
1	0	0
1	1	1

Exercice 3 : On donne la table de vérité d'un opérateur MACHIN :

Donner une expression logique pouvant correspondre à cette table de vérité.





2 Les portes logiques

2.1 Principe

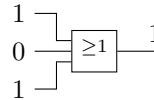
En électronique digitale, les opérations logiques sont effectuées par des portes logiques. Ce sont des circuits qui combinent les signaux logiques présentés à leurs entrées sous forme de tensions. On aura par exemple 5V pour représenter l'état logique 1 et 0V pour représenter l'état 0. Ces portes logiques sont réalisées en partie grâce à des transistors³. C'est bien simple : sans portes logiques, il n'y aurait ni électronique, ni informatique aujourd'hui.

1. Les montages avec transistors pourront être réalisés avec un Arduino dans le cadre d'un TP.
 2. Savoir : opérations logiques
 3. Il pourra être intéressant à ce stade de faire une démonstration avec un montage Arduino avec deux boutons poussoirs et une ampoule pour simuler la porte logique OR

Portes logiques usuelles :

Opérateur	NOT	AND	OR	XOR
Symbole				

A part la porte logique NOT, les autres portes logiques ne sont pas limitées à deux entrées, mais peuvent en prendre plusieurs⁴.



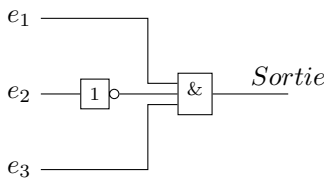
Exemple : Avec une porte logique OR à trois entrées :

2.2 Comparateurs

Comment tester électroniquement qu'un nombre est égal à une constante ? Par exemple $x == 5$?

Commençons par l'écriture binaire du nombre : 5 s'écrit 101. Nous avons un nombre codé sur 3 bits.

Si x est codé sur 3 bits, chacun des bits peut être assimilé à une entrée : $e_1e_2e_3$. L'astuce consiste à utiliser des portes NOT, puis une porte AND.



Exercice 4 : Construire la table de vérité du circuit ci-dessus :

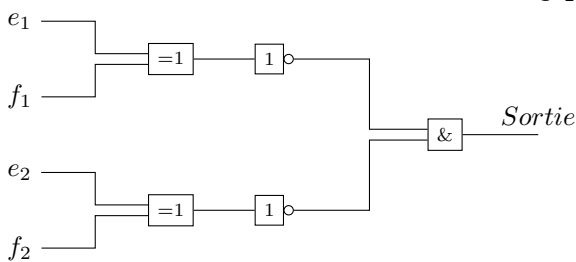
e_1	e_2	e_3	Sortie
0	0	0	...
...

On a donc un circuit qui renvoie un booléen : 1 si x est égal à 5, 0 sinon.

Exercice 5 : Dans le même esprit, proposer un circuit qui teste : $x == 6$

Exercice 6 : Comment maintenant comparer deux nombres ? Par exemple comment tester $x == y$?

Prenons deux nombres codés sur 2 bits : $x = e_1e_2$ et $y = f_1f_2$. Voici un circuit :




Montrer que ce circuit logique compare bien deux nombres codés sur 2 bits.

Exercice 7 : Un comparateur n bits prend en entrée deux nombres de n bits (soit $2n$ bits au total) et possède une sortie unique qui doit être 1 si les deux nombres sont égaux, 0 sinon. Construire un circuit logique qui implémente un comparateur 3 bits.

2.3 Circuits combinatoires

Exercice 8 : L'opérateur **NOR** est un opérateur logique binaire correspondant à **NOT OR**.

- Donner la table de vérité de cet opérateur.
- Montrer que les quatre opérateurs logiques usuels (NOT, AND, OR, XOR) peuvent être construits uniquement à l'aide de cet opérateur.
- Implémenter ces quatre opérateurs sous forme de circuit logique utilisant uniquement des portes NOR. Le symbole de la porte NOR est : 

Exercice 9 : La porte logique XOR existe en version avec plusieurs entrées. Par exemple avec une version à 3 entrées e_1, e_2, e_3 , elle correspond à l'opérateur logique : $(e_1 \text{ XOR } e_2) \text{ XOR } e_3$. Proposer un circuit logique, n'utilisant que des portes XOR à deux entrées, pouvant représenter une porte XOR à 3 entrées. En donner la table de vérité.

Faire de même avec une porte XOR à 4 entrées. Que peut-on conjecturer sur le comportement d'une porte XOR à n entrées ?

⁴ Cas particulier : la porte logique XOR à plusieurs entrées conduit à un bit de parité