CALCUL BOOLEEN.

1. Introduction : les portes logiques au cœur de l'ordinateur.

   Dans ce chapitre, nous allons comprendre comment des outils mathématiques permettent de concevoir des circuits électroniques élémentaires, comme celui qui permet l'adition de deux nombres. Nous verrons dans le chapitre « architecture » comment ces circuits sont ensuite intégrés dans l'unité centrale (entre autres) d'un ordinateur.

   On se limitera à quelques exemples très simples (mais réalistes, voir réels). Néanmoins, les outils et méthodes utilisés actuellement sont ceux qui seront présentés ci-après. Nous allons donc voir dans un premier temps les portes logiques, qui sont les circuits électroniques utilisés aussi bien pour fabriquer l'unité arithmétique et logique, que la mémoire, ou encore le contrôleur de bus (qui n'est pas le gars avec la casquette dans le tram, mais l'ensemble de circuits permettant à l'UAL de « réclamer » la bonne donnée en mémoire).

   Remarquons que l'on peut encore descendre d'un étage, jusqu'aux transistors, qui permettent de fabriquer les portes. Cette étape supplémentaire n'est pas très complexe, et peut intéresser tous ceux qui ne sont pas rebutés par la physique ; cela demande de ne pas être totalement allergique à l'électricité. Nous n'aurons pas le temps de faire cette étape supplémentaire, vous trouverez de quoi nourrir votre curiosité ici :

   - Le plus simple peut-être, notamment à partir de l'intertitre « transistors » : http://www.liafa.jussieu.fr/~carton/Enseignement/Architecture/Cours/Gates/
   - un cours qui va assez loin, plus que ce que l'on fait. Le début est très clair sur le fonctionnement des transistors www.ai.univ-paris8.fr/~audibert/ens/6-**PORTESLOGIQUES**.pdf

   Si vous souhaitez faire cet approfondissement, je vous conseille de ne le faire qu'après avoir vu ce chapitre, pour que le vocabulaire vous soit plus accessible.

   Pour comprendre les bases du calcul booléen :

   - Supposons que votre ordinateur de bureau soit branché sur le secteur à l'aide d'une prise à interrupteur. Pour tchatter, il faut allumer l'ordinateur **ET** la prise secteur.
   - Supposons que pris d'une fringale à deux heures du matin, vous vous leviez pour aller piller le frigo. Pour satisfaire votre envie, vous mangez un yaourt au chocolat **OU** de la glace à la vanille (« ou » au sens large !). Par contre, pour allumer la lumière dans le couloir à l'aide d'un des deux interrupteurs, est-ce le même **OU** que vous appliquez aux interrupteurs ?
   - Supposons que vous discutiez du dernier James Bond avec un ami qui n'est pas du tout de votre avis : c'est le **NON** que vous utiliserez.

2. Calcul booléen.

   Dans un circuit électronique, on considère qu'il n'y a que deux niveaux de tension possible : « haut » et « bas ». Ces deux niveau seront notés 1 et 0. Ce choix est arbitraire, on pourrait noter vrai/faux, marche/arrêt, positif/négatif, $\top/\bot$ (truc/antitruc) voire table/chaise…

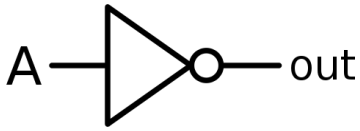
   Le calcul booléen porte uniquement sur ces valeurs logiques 0 et 1. Le résultat d'un calcul sera forcément soit 0, soit 1.

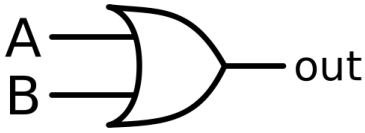
   Remarque : Ce **n'est pas** du calcul en binaire. Le binaire est une autre notation des nombres, qui permet d'obtenir n'importe quel résultat numérique. Pour bien différencier le binaire du calcul booléen, on peut constater que le résultat d'un calcul booléen est « truc » ou « antitruc », ce qui n'a pas grand-chose à voir avec un nombre.

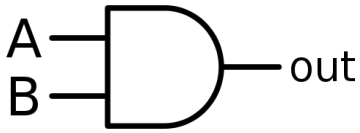
   Table d'adition (OU) et de multiplication (ET) :

Opérateurs booléens de base.

<table>
<tr><th>Type, symbole opératoire</th><th>Symbole de la porte logique correspondante (américain)</th><th>Table de vérité</th></tr>
<tr><td>NON

$\overline{A}$, non-A , $\neg A$</td><td> 
</td><td>
<table>
<tr><th>Entrée</th><th>Sortie</th></tr>
<tr><td>A</td><td>$\overline{A}$</td></tr>
<tr><td>0</td><td>1</td></tr>
<tr><td>1</td><td>0</td></tr>
</table>
</td></tr>
<tr><td>OU

A+B , $A \vee B$, $A|B$</td><td> 
</td><td>
<table>
<tr><th colspan="2">Entrées</th><th>Sortie</th></tr>
<tr><td>A</td><td>B</td><td>A+B</td></tr>
<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>
</table>
</td></tr>
<tr><td>ET

$A \cdot B$, $A \wedge B$, A&B</td><td> 
</td><td>
<table>
<tr><th colspan="2">Entrées</th><th>Sortie</th></tr>
<tr><td>A</td><td>B</td><td>$A \cdot B$</td></tr>
<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>
</table>
</td></tr>
</table>

3. <u>Propriétés du calcul booléen (ne sont pas à connaître par cœur, juste à comprendre).</u>

Propriétés usuelles :

- Associativité ; les parenthèses sont inutiles lorsque l'on effectue plusieurs fois la même opération.
  $(a+b)+c = a+(b+c) = a+b+c$ et $(a \cdot b) \cdot c = a \cdot (b \cdot c) = a \cdot b \cdot c$
- Commutativité (l'ordre n'a pas d'importance)
  $a+b = b+a$ et $a \cdot b = b \cdot a$
- Éléments neutres : $a+0=0$ et $a \cdot 1 = a$
- Distributivité du produit par rapport à l'addition: $a \cdot (b+c) = a \cdot b + a \cdot c$

Les propriétés que l'on peut inférer à partir d'exemples réalistes :

- Complémentarité : $\overline{\overline{a}} = a$ $a + \overline{a} = 1$ $a \cdot \overline{a} = 0$
- Absorption : $0 \cdot a = 0$ $1 + a = 1$ (d'où $b + b \cdot a = b$)

Des propriétés encore plus surprenantes :

- Idempotence : $a+a=a$ $a \cdot a = a$ d'où $a+a+\ldots+a=a$ et $a \cdot a \cdot \ldots \cdot a = a$
- Simplification : $a + \overline{a} \cdot b = a + b$
- Redondance : $a \cdot b + \overline{a} \cdot c = a \cdot b + \overline{a} \cdot c + b \cdot c$
- Distributivité de la somme par rapport au produit : $a + (b \cdot c) = (a+b) \cdot (a+c)$

Lois de Morgan : $\overline{a+b} = \overline{a} \cdot \overline{b}$ et $\overline{a \cdot b} = \overline{a} + \overline{b}$

Preuves de certaines de ces propriétés à faire en exercice (distributivités, $b + b \cdot a = b$ par deux méthodes, idempotence, simplification, redondance, lois de Morgan)

<u>Propriétés et implémentation du calcul booléen en Python</u>

Les opérateurs booléens sont `and`, `or`, et `not`.

L'évaluation de ces opérateurs est dite séquentielle (ou fainéante ☺), c'est-à-dire qu'elle se fait de la gauche vers la droite, sans forcément tout calculer. On stoppe l'évaluation de la condition dès que l'on a l'assurance qu'elle est vraie (ou fausse). Comme on arrête cette évaluation dès que possible, on parle d' « évaluation paresseuse ».

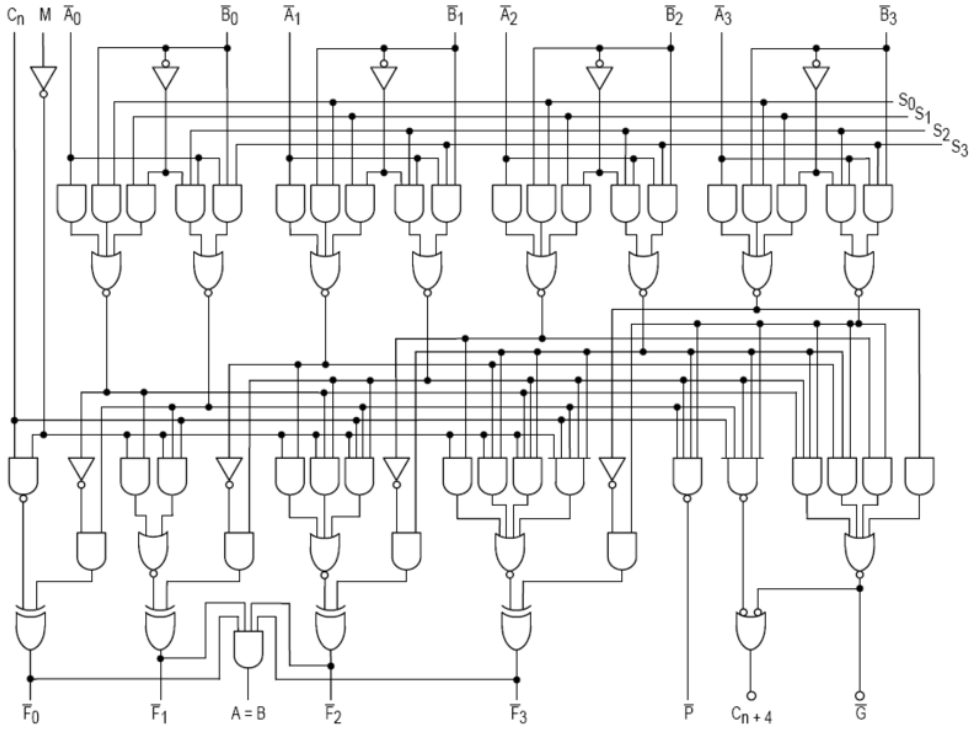
Exemples :

- A et B est faux dès que A est faux, il est inutile de calculer B
- A ou B est vrai dès que A est vrai, il est inutile de calculer B
- Cela permet de construire des tests comme suit sans provoquer d'erreur :
  - `while x != 0 and 1/x < 1e-10 :`
    `...`
    Si $x = 0$, le test ne cherchera pas à calculer $1/x$ qui donnerait une erreur du type division par 0
  - `if i >= N or t[i] > A : # pour un tableau de taille N`
    `...`
    Si par exemple $i = N + 1$, le test ne calculera pas `t[N + 1]`, qui donnerait une erreur detype « list index out of range »

Remarque personnelle : ceci peut donner lieu à des dérives de programmation crasseuse plus que pareseuse… il faut savoir que cela existe. A titre subjectif l'usage de « l'évaluation paresseuse » est au mieux à fortement limiter, et à expliquer/commenter lorsque l'on s'en sert.

4. Applications.

En hors d'œuvre un peu copieux, voire indigeste, voici un exemple d'UAL (UAL 4 bits 74181, utilisé dans un des tous premiers micro-ordinateurs en 1973). Cette UAL peut effectuer $2^4 \times 3 = 48$ opérations. Le codage de la fonction se fait sur S, de 0000 à 1111 sur les entrées S_0 S_1 S_2 S_3 (soit 16 possibilités), et sur M qui vaut 0 ou 1. Si M vaut 0, l'entrée C_n double les choix, suivant sa valeur 0 ou 1. Les entrées A et B ont 4 bits chacune, en A_0 A_1 A_2 A_3 et B_0 B_1 B_2 B_3. La sortie est F, en F_0 F_1 F_2 F_3.
Le tableau des fonctions est sur http://fr.wikipedia.org/wiki/Arithmetic_and_logical_unit, il comporte peut-être des erreurs que vous aurez à cœur de modifier…



Nous allons faire plus simple : un additionneur un bit. Le but est de construire le circuit permettant l'addition de deux bits, l'un étant à l'entrée A et l'autre à l'entrée B. Ce circuit doit pouvoir être utilisé pour additionner des nombres en binaires, comme nous l'avons fait dans le chapitre sur le codage. Pour rappel, on a vu que lorsque l'on ajoutait deux nombres :

- on les ajoute chiffre à chiffre, donc en binaire bit à bit ;
- il peut y avoir des retenues.

Commençons par additionner deux bits de poids faible, c'est à dire le plus à droite, sans retenue.
On appelle A et B les deux bits que l'on ajoute, S la somme et C la retenue (carry in english).

On a alors $A + B = \overline{CS_2}$, la table de vérité suivante donne les quatre possibilités.

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

On constate que les expressions booléennes de C et de S sont et .
D'où la construction du circuit (dont le petit nom est semi-additionneur) ci-dessous :

Passons au cas général (additionneur complet 2 bits)
On ajoute cette fois-ci deux bits A et B, une retenue entrante C_0 (carry in). On obtient une retenue sortante C_1 (carry out) et la somme S.

On a alors $A + B + C_0 = \overline{C_1 S_2}$, la table de vérité suivante donne les huit possibilités.

A	B	C_0	C_1	S
0	0			
0	0			
0	1			
0	1			
1	0			
1	0			
1	1			
1	1			

D'où les expressions booléennes :

formules (1) et (1b)
formules (2) et (2b)

Remarque : on peut trouver la deuxième formule d'une autre façon, en remarquant qu'il y a retenue sortante à partir du moment où au moins deux des bits d'entrée sont égaux à 1.
D'où (2t)

D'où le circuit, sous deux formes - on peut en faire d'autres - :

Avec les formules (1) et (2t)	
Avec les formules (1b) et (2b)	

Une remarque finale : il y a des gens qui se sont amusés à construire ça sous Minecraft… et qui sont même allés jusqu'aux calculatrices complètes (additions/soustractions à 6 chiffres, multiplications à 3 chiffres, fonctions trigonométriques, tracé de courbes… A l'échelle de Minecraft, ça fait 5 millions de blocs).

EXERCICES CALCUL BOOLEEN ET APPLICATIONS

Ex 1.

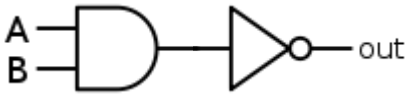
A l'aide des tables de vérité, montrer l'égalité des expressions booléennes suivantes : $\overline{a+b} = \overline{a} \cdot \overline{b}$

C'est la première loi de Morgan.

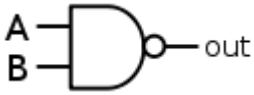
Montrer la deuxième loi de Morgan : $\overline{a \cdot b} = \overline{a} + \overline{b}$

Ex 2.

Donner la table de vérité et l'expression booléenne de la porte logique :

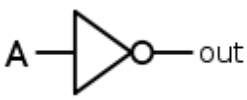


On représente cette porte par le symbole



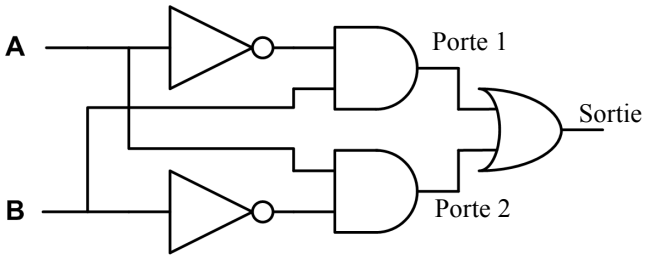
. On l'appelle :

En effet le petit cercle ○ en sortie symbolise la négation, il remplace



Ex 2.

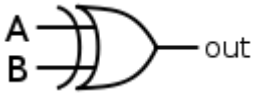
Compléter la table de vérité de la porte logique suivante, et donner son expression booléenne.



A	B	$\overline{A}$	$\overline{B}$	Porte 1	Porte 2	Sortie
0	0					
0	1					
1	0					
1	1					

A quoi correspond cette porte « en français » ?

Donner une expression booléenne pour cette porte.

Elle s'appelle . On la représente par 

Elle nous sera très utile pour la suite du cours.

Ex 4 : le multiplexeur.

Pour comprendre le principe, prenons un exemple. Dans la gare de Montpellier, on suppose qu'il y a 8 quais, numérotés de 0 à 7, et que 8 trains sont à quai, tous prêts à partir, tous vers Narbonne. Or il n'y a qu'une seule voie qui va vers Narbonne. Si les trains partent tous en même temps, cela risque de poser quelques problèmes… Il faut donc, à l'aide d'un aiguillage, choisir le train qui va partir en premier. C'est le rôle du multiplexeur : il sélectionne une entrée parmi plusieurs à l'aide de son numéro. Pour 8 voies/entrées, comme $8 = 2^3$, on codera la sélection S de l'entrée sur 3 bits. Toujours dans l'exemple, si S = 101, alors c'est le train sur la voie 5 qui partira.

Nous allons nous contenter d'un multiplexeur à 1 bit, qui peut donc choisir entre deux entrées A_0 et A_1. Si le bit de sélection S est à 0, c'est la porte A_0 qui sera sélectionnée. Si le bit S est à 1, c'est la porte A_1 qui sera sélectionnée. Il y a donc trois entrées : A_0, A_1, et S. Et une sortie, qui est égale à la valeur de l'entrée sélectionnée.

Entrées			
S	A_0	A_1	Sortie
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

1. Complétez la table de vérité du multiplexeur à 1 bit ci-contre.
2. Donner l'expression booléenne du multiplexeur 1 bit sous la forme « $Sortie = Machin \cdot Truc + Bidule \cdot zaffair$ ». Machin, truc, bidule, zaffair, sont à remplacer par la bonne entrée A_0, A_1, S ou par leur négation. Remarque : il suffit de trouver la formule pour le cas Sortie = 1.
3. En déduire un circuit logique donnant le multiplexeur à 1 bit.

Ex 5 : le décodeur.

C'est en quelque sorte l'inverse du multiplexeur.

Ce circuit « va chercher » la case mémoire demandée en entrée. Si l'entrée est codée sur 1 bit, ce circuit va pouvoir distinguer 2 cases mémoires, celle d'adresse 0 et celle d'adresse 1. Si l'entrée est codée sur 2 bits, le décodeur pourra distinguer 4 cases mémoires, adressées de 00 = 0 à 11 = 3. Si l'entrée est codée sur n bits, le décodeur pourra distinguer entre 2^n cases mémoires, adressées de 0 à 111…1 = 2^{n-1}. Donc un décodeur à n bits possède n entrées, notées A_0, A_1, …, A_{n-1} ; et 2^{n-1} sorties, notées S_0, S_1, … , $S_{2^{n-1}}$.

Entrée	Sorties	
A	S_0	S_1
0	1	0
1	0	1

La sortie sélectionnée sera à 1, les autres à 0.

1. Décodeur à 1 bit.

   La table de vérité du décodeur à 1 bit est donnée ci-dessus.
   Donner l'expression booléenne de S_0 en fonction de A (ou de sa négation), puis celle de S_1.
   En déduire un circuit logique donnant le décodeur à 1 bit.

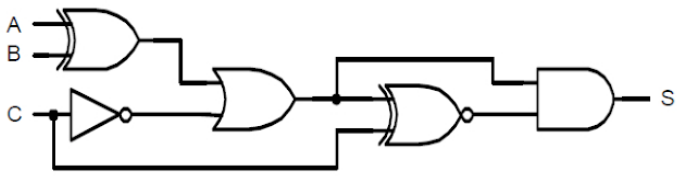
Entrées		Sorties			
A_0	A_1	S_0	S_1	S_2	S_3

2. Décodeur à 2 bits.
   Compléter la table de vérité du décodeur à 2 bits.
   Donner l'expression booléenne pour chaque sortie.
   En déduire un circuit logique donnant le décodeur à 2 bits.

Ex 6.

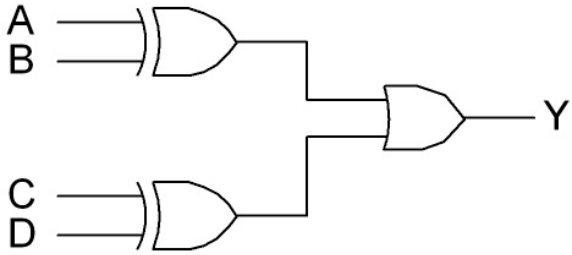
Que font ces portes ? On donnera la table de vérité et l'expression booléenne correspondante

Porte 1



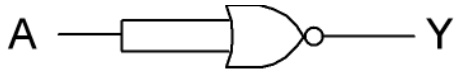
A, B et C en entrée, S en sortie

Porte 2



A, B, C et D en entrée, Y en sortie

Porte 3



A en entrée, Y en sortie

Ex 7.

Démontrer ou simplifier les expressions booléennes suivantes à l'aide des propriétés données dans le cours.

$a+\overline{b}\cdot a=1$ $\quad \overline{a\cdot b+\overline{a}+\overline{b}}=0$ $\quad a\cdot(b+c)=a\cdot b\cdot c+a\cdot\overline{b}\cdot c+a\cdot b\cdot\overline{c}$ $\quad \overline{a}\cdot(a+\overline{b})\cdot(\overline{a}+b)=\overline{a}\cdot\overline{b}$

$(b+a\cdot b+c)\cdot(a+\overline{b}+\overline{a}\cdot\overline{c})=a\cdot b+b\cdot\overline{c}+\overline{b}\cdot c$ $\quad a\cdot\overline{c}+\overline{a}\cdot b\cdot c+a\cdot b=a\cdot\overline{c}+b\cdot c$

$A=a\cdot b+a\cdot\overline{b}+\overline{a}\cdot\overline{b}$ $\quad B=\left(\overline{\overline{a}+\overline{d}}\right)\cdot\left(\overline{a\cdot b+a\cdot\overline{b}\cdot\overline{d}}\right)$ (une seule lettre !) $\quad C=\overline{a}\cdot b\cdot c+a\cdot c+a\cdot\overline{b}\cdot\overline{c}+\overline{a}\cdot\overline{b}$

Ex 8.

Reprendre le circuit de l'additionneur 2 bits. En simplifier le circuit, en utilisant une porte « multiplexeur » (les 3 entrées seront les bits A et B, et la retenue entrante C_0). On pourra créer une porte « mux » pour simplifier
Construire un additionneur à 4 bits, qui ajoute deux nombres de deux bits (4 entrées et 3 sorties). On pourra créer une porte « add », pour simplifier (qui admet en entrée deux bits A et B, C_0 et donne S et C_1 en sortie).

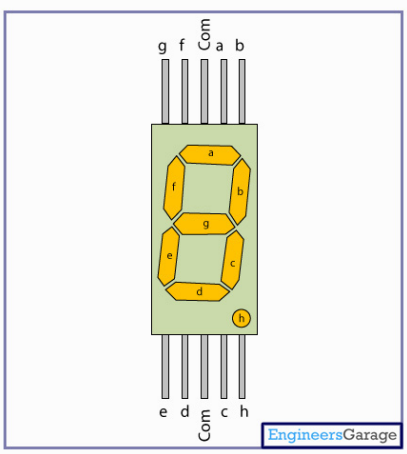
Ex 9.

Construire un circuit à 9 (A_0, A_1, …, A_7, et D) entrées et 8 sorties (B_0, B_1,…, B_7) telles que :

- Si D = 0, le circuit ne fait rien : on a $B_i = A_i$ pour *i* compris entre 1 et 7
- Si D = 1, le circuit multiplie A par 2 en décalant à gauche les bits (cf. cours sur le codage) : on a $B_i = A_{i-1}$ pour *i* compris entre 1 et 7. Que devient B_0 ?

Ex 10.

On veut afficher sur un « afficheur 7 segment » les chiffres 0, 1, 2 et 3 en allumant les bons segments (schéma ci-dessous).
Donner la table de vérité des 8 sorties correspondantes respectivement aux segments *a* à *g* (on remarque qu'ici *h* n'est pas utilisé).



Ex 10b.

Reprendre l'exercice précédent pour un affichage de tous les chiffres en hexadécimal (A, C, E et F sont en majuscules, b et d en minuscules).