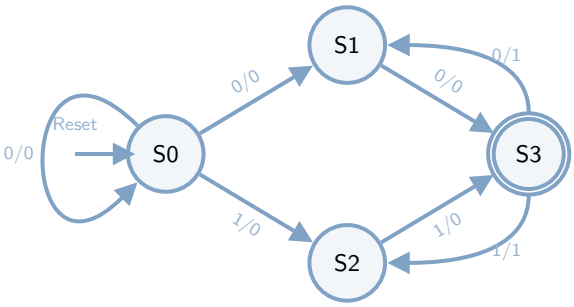
# Introduction à l' Electronique

## NUMÉRIQUE

Travaux Dirigés – 1ère année

**Jean-Christophe Le Lann**

**Enoncés des TD**




Année académique 2026-2027

Version 2.0

# TD 1 – Électronique Numérique

Représentation des nombres et Logique combinatoire

**Notes et Conseils**

- Le premier objectif de ce TD est de manipuler différentes représentations des *nombres* et notamment de maîtriser les changements de base **sans calculatrice**.
- En dehors du TD, il est suggéré de vérifier que vous êtes suffisamment adroit pour les réaliser également *avec* calculatrice ou à l'aide de votre langage de programmation préféré.
- Le second objectif est de se familiariser avec l'algèbre de Boole.
- *Note : Tous les exercices ne seront pas nécessairement corrigés lors des séances de TD.*

## Première partie

# Représentation des nombres

## 1 Compter en base 2 et base 16

1. Compter de 0 à 15 en binaire, puis en hexadécimal. *(On conservera ces valeurs dans un tableau pour la durée de cette partie du TD)*
2. Écrire les puissances de $2^n$ pour $n$ variant de 0 à 10. En déduire une valeur approchée de $\log_{10} 2$, puis $\log_{10} 5$. Avec un procédé similaire, trouver une valeur approchée de $\log_{10} 3$.
3. Calculer la valeur décimale de $10101011_2$ en passant par les puissances de 2, puis en passant par les puissances de 16.
4. Écrire les puissances de $2^{-n}$ pour $n$ variant de 1 à 4.

*Remarque :* Des nombres possédant une partie décimale sont transformés ici en de simples *entiers*, en supposant que la position de la virgule est fixée à l'avance. Cette représentation est appelée **virgule fixe** (*fixed-point*). Elle est très rapide à traiter par l'électronique, à coût matériel très faible, et est très utilisée en traitement du signal embarqué (DSP). À l'inverse, une représentation en **virgule flottante** est plus consommatrice de ressources (surface) et de temps, mais offre une plage dynamique et une précision beaucoup plus importantes (norme IEEE 754 traitée dans l'exercice 4).

## 2 Conversions entre bases

1. Convertir le nombre $39_{10}$ en binaire.
2. Convertir le nombre $(0,375)_{10}$ en binaire.
3. En déduire la représentation binaire du nombre $(39,125)_{10}$.
4. Convertir le nombre $(0,2)_{10}$ en binaire. Que remarquez-vous ?

5. Convertir $(345,158)_{10}$ en octal.
6. Convertir $(389)_{10}$ en hexadécimal.
7. Convertir $011001111011101_2$ en hexadécimal.
8. Convertir $(10110001101011,1111)_2$ en octal.
9. Convertir $(10110001101011,111101)_2$ en hexadécimal.
10. Le nombre $101110001_2$ est écrit en BCD (décimal codé binaire). Convertir le nombre décimal obtenu en binaire pur. Combien de bits sont alors nécessaires ? Conclure.

## 3 Nombres signés en complément à 2

### Rappels

Jusqu'ici, nous nous sommes intéressés aux nombres positifs. Il existe plusieurs manières de représenter un nombre négatif $n$. La manière la plus intuitive serait de dédier un bit (le plus à gauche) pour préciser le signe (ex : 0 pour +, 1 pour −). Malheureusement, cette représentation possède deux défauts majeurs : il existe deux représentations du zéro (+0 et −0) et il faut modifier l'algorithme d'addition.

Les architectes des ordinateurs ont retenu une autre représentation plus pratique : le **complément à 2**. C'est cette représentation qui régit le fonctionnement des *unités arithmétiques et logiques* (ALU) des ordinateurs.

Avec $k$ bits, on peut représenter l'ensemble des nombres entiers dans l'intervalle :

$$[-2^{k-1},\ \ldots,\ 2^{k-1}-1]$$

Le complément à 2 d'un entier $n$ est obtenu par le calcul de la quantité $2^k - n$. On peut vérifier aisément que :

— Le complément à 2 du complément à 2 d'un entier $n$ est $n$ lui-même.
— Le complément à 2 de zéro est zéro (on néglige la retenue qui dépasse la taille $k$ fixée).

On doit retenir la formule de passage en complément à 2 d'un nombre négatif $-X$ :

$$-X = \overline{X} + 1 \tag{1}$$

La barre figurant au-dessus du $X$ signifie "inversion de tous les bits" (complément à 1). Si l'entier $n$ est codé sur $k$ bits $(s_{k-1} \ldots s_0)$ en complément à 2, sa valeur décimale est :

$$n_{10} = -s_{k-1}2^{k-1} + \sum_{i=0}^{k-2} s_i 2^i \tag{2}$$

### Exercices

1. Déterminer la plage des valeurs des entiers signés sur $k = 4$ bits. Représenter ces valeurs en binaire. Quelles sont les représentations des valeurs extrêmes ? de 0, 1, et -1 ?
2. Quel est le complément à 2 du nombre 8, codé sur 6 bits ? Utiliser la formule (1) et vérifier le résultat en appliquant la formule (2).
3. Soustraire : $91_{10} - 108_{10}$ en passant par le complément à 2.
4. Écrire les nombres suivants en binaire sur 6 bits : $+24, +31, -24, -31$. Passez ensuite à 7 bits. Que remarquez-vous ?
5. Soustraire : $1100\ 0101\ 0100\ 0000_2 - 101000\ 1111\ 1111_2 - 110\ 1111\ 1110_2 - 10\ 1110_2$.

## 4 Norme IEEE 754

**Rappel : norme IEEE 754 (simple précision, 32 bits)** Un flottant est codé par trois champs : 1 bit de signe $s$, 8 bits d'exposant $e$ et 23 bits de mantisse $m$. La valeur représentée est, pour la plupart des nombres :

$$V = (-1)^s \times 2^{e-127} \times (1+m)$$

où $m = \sum_{i=1}^{23} b_i 2^{-i}$ est la partie fractionnaire. L'exposant est représenté en excédent (biais) à 127. La mantisse est toujours normalisée avec un 1 implicite (gain de place), sauf pour les cas particuliers suivants :

- si $e = 255$ et $m = 0$, alors $V = \pm\infty$ (selon $s$) ;
- si $e = 255$ et $m \neq 0$, alors $V =$ `NaN` (Not a Number) ;
- si $e = 0$ et $m = 0$, alors $V = \pm 0$ ;
- si $e = 0$ et $m \neq 0$, il s'agit d'un nombre dénormalisé, dont la valeur est $V = (-1)^s \times 2^{-126} \times m$.

**Exercice :** Représenter le nombre $238,4375$ dans sa représentation IEEE 754 (32 bits) en virgule flottante.

# Deuxième partie
# Algèbre de Boole et Logique combinatoire

Nous commençons par quelques manipulation élementaires autour de la minimisation d'expressions booléennes et leur mise en forme canonique, à l'aide de l'algèbre de Boole. Nous illustrons également la méthode des tableaux de Karnaugh qu'il faut voir comme une technique tabulée de minimisation astucieuse qu'il est intéressant de connaître pour de petits circuits. Nous poursuivons par la construction d'opérateurs matériels élémentaires : additionneurs, multiplexeurs, comparateurs.

## 5 Quelques formulations booléennes

Écrire sous la première forme canonique (somme de mintermes) les fonctions définies par les propositions suivantes :

1. $F(a,b,c) = 1$ si et seulement si aucune des variables $a, b, c$ ne prend la valeur 1.
2. $F(a,b,c) = 1$ si et seulement si au plus une des variables $a, b, c$ prend la valeur 0.

## 6 Minimisation d'expression booléennes

Simplifier algébriquement les fonctions suivantes :

1. $F = \overline{x}\overline{y} + xy + \overline{x}y$
2. $F = (x + \overline{y})(x\overline{y} + z)z$

## 7 Simplifications par tableaux de Karnaugh

Simplifier, par la méthode des tableaux de Karnaugh, les fonctions booléennes suivantes. Dessiner le circuit correspondant et déterminer son chemin critique, en supposant que toutes les portes présentent un temps de propagation de 5 ns.

### 7.1 K-map à 3 variables

1. $F(a,b,c) = \overline{a}\overline{b}c + \overline{a}b\overline{c} + ab\overline{c}$
2. $F(a,b,c) = \overline{a}b\overline{c} + \overline{a}bc + ab\overline{c}$
3. $F(a,b,c) = \overline{a}\overline{b}c + a\overline{b}\overline{c} + ab\overline{c}$, sachant que la valeur de $F$ pour les combinaisons $\overline{a}bc$ et $abc$ est indifférente (donc care $X$).

### 7.2 K-map à 4 variables

1. $F(a,b,c,d) = \overline{a}b\overline{c}\overline{d} + \overline{a}b\overline{c}d + \overline{a}bcd + \overline{a}bc\overline{d}$
2. $F(a,b,c,d) = \overline{a}\overline{b}\overline{c}d + \overline{a}\overline{b}cd + a\overline{b}\overline{c}d + a\overline{b}cd$

## 8 Équations logiques de circuits de base

### 8.1 Additionneur

1. Établir la table de vérité d'un *demi-additionneur* logique, puis les deux équations logiques associées : ce circuit additionne simplement 2 entrées booléennes $a$ et $b$ (donc *sans retenue entrante*). Il calcule deux sorties : la somme $s$ et la retenue sortante $c_o$. Dessiner le circuit.
2. Établir la table de vérité d'un *additionneur complet* pour deux opérandes 1 bit. Ce dernier tient désormais compte d'une retenue entrante $c_i$. Établir les équations simplifiées des sorties, puis dessiner le circuit correspondant. Au préalable, on démontrera que : $\overline{x \oplus y} = x \oplus \overline{y}$.
3. Montrer qu'un additionneur complet "1 bit" peut réutiliser le demi-additionneur de la question 1.
4. Dessiner un additionneur 4 bits en réutilisant l'additionneur 1 bit.
5. Quel est le chemin critique de ce dernier circuit ? On rappelle que le chemin critique est le parcours le plus long entre une entrée et une sortie combinatoire. On le mesurera en nombre de portes logiques élémentaires traversées.
6. Peut-on imaginer d'anticiper le calcul de la retenue ? Si oui, comment ?

### 8.2 Multiplexeur

La fonction d'un multiplexeur est de sélectionner, grâce à un signal de contrôle, un signal d'entrée parmi plusieurs et de transmettre sa valeur sur une sortie unique.

1. Dessiner le symbole communément admis pour représenter un multiplexeur à 2 entrées binaires (chacune codée sur 1 bit). On parle de multiplexeur "2 vers 1".
2. Calculer l'équation d'un multiplexeur "2 vers 1" pour des entrées codées sur 1 bit.
3. Lorsque les données d'entrées sont codées sur $n > 1$ bits, chacun des bits du signal multiplexé suit évidemment un chemin similaire. Dessiner un tel multiplexeur à 2 entrées, pour $n = 2$.
4. Etablir la structure du multiplexeur à $m$ entrées, à l'aide de multiplexeurs.
5. Calculer le nombre de couches logiques traversées pour un multiplexeur $m, n$ ("m vers 1" et dont les données sont codées sur $n$ bits).

### 8.3 Comparateur

Soit deux données $a$ et $b$ quelconques, encodées par deux variables booléennes générales $(a_{n-1}, \ldots, a_0) \in \mathcal{B}^n$ et $(b_{n-1}, \ldots, b_0) \in \mathcal{B}^n$.

1. Démontrer que le résultat booléen $f$ de comparaison d'égalité entre les deux données vaut :
$$f(a,b) = \prod_{i=0}^{i=n} \overline{a_i \oplus b_i}$$
2. Dessiner la structure du comparateur matériel pour $n = 4$.

## 9 Formes canoniques

**Prérequis : Codage binaire et indexation des combinaisons booléennes**
Pour un système à $n$ variables logiques ordonnées, il existe $2^n$ combinaisons possibles de valeurs. Chaque combinaison peut être interprétée comme un nombre binaire, ce qui permet de lui associer de manière unique un indice entier compris entre 0 et $2^n - 1$. Par exemple, pour trois variables ordonnées $(x, y, z)$, la combinaison de vérités $(1, 0, 1)$ est associée au mot binaire $101_2$, qui correspond à l'indice décimal 5 : ce dernier pourra être évoqué comme le monôme $m_5$. Ce formalisme d'indexation numérique standardise la désignation de n'importe quel état du système et s'applique de manière générale en logique combinatoire, notamment pour l'organisation des tables de vérité, le référencement des cases d'un tableau de Karnaugh, etc.

**Formes canoniques et passage de l'une à l'autre**
La **première forme canonique** (ou Forme Normale Disjonctive, FND) est une somme de produits appelés mintermes $(m_i)$, où chaque $m_i$ est le produit logique associé à la combinaison d'indice $i$ pour laquelle la fonction vaut 1. À l'inverse, la **deuxième forme canonique** (ou Forme Normale Conjonctive, FNC) est un produit de sommes appelées maxterms $(M_i)$, où chaque $M_i$ est la somme logique associée à la combinaison d'indice $i$ pour laquelle la fonction vaut 0. Le passage d'une forme à l'autre exploite directement la complémentarité des indices au sein de l'ensemble des $2^n$ combinaisons. La fonction inverse $\bar{f}$ étant constituée des mintermes dont les indices sont absents de $f$, l'application de la double négation $(f = \overline{\bar{f}})$ et des lois de De Morgan permet de transformer la somme de ces mintermes manquants en un produit de maxterms de mêmes indices :

$$\begin{aligned} f(x,y,z) &= m_1 + m_3 + m_5 \quad \text{(la fonction vaut 1 pour les indices 1, 3 et 5)} \\ \bar{f}(x,y,z) &= m_0 + m_2 + m_4 + m_6 + m_7 \quad \text{(indices restants parmi les } 2^3 \text{ possibles)} \\ f(x,y,z) = \overline{\bar{f}(x,y,z)} &= M_0 \cdot M_2 \cdot M_4 \cdot M_6 \cdot M_7 \quad \text{(par passage au dual via De Morgan)} \end{aligned}$$

Ainsi, commuter entre les deux formes canoniques revient simplement à inverser la nature de l'opérateur principal (somme ou produit) en basculant sur l'ensemble complémentaire des indices de l'univers des combinaisons. *Note : Ces formes ne sont pas simplifiées.*

Établir les deux formes canoniques des fonctions suivantes :

1. $F_1 = xy + yz + xz$
2. $F_2 = x + yz + \overline{yz}t$

## 10 Vu-mètre

On se propose de réaliser un “contrôleur de vu-mètre” : c’est un dispositif qui permet de visualiser une grandeur numérique (un volume sonore par exemple) au moyen d’une barrette lumineuse qui s’étire plus ou moins en fonction de la grandeur.

1. Schématiser le principe d’ensemble dans le cas où la grandeur à visualiser est codée sur 3 bits.
2. À partir du tableau de Karnaugh, trouver une réalisation possible.
3. On suppose que l’on dispose désormais d’un composant complexe de décodage (un décodeur 3 vers 8). Rappeler la table de vérité de ce circuit. Utiliser alors ce décodeur pour proposer une nouvelle réalisation.

## 11 Le jeu des cambrioleurs

Le principe du jeu [1] est le suivant : un jeton est introduit par le joueur en haut du dispositif. Le but est de faire circuler ce jeton dans le dispositif, jusqu’à ce qu’il apparaisse en sortie. Dans l’intervalle, le jeton peut rester coincé à différents endroits (ou “niveaux”) : seul un code secret correctement introduit à cet endroit permet au jeton de continuer son parcours. Sinon, l’utilisateur voit le code d’erreur `0xDEAD` apparaître en sortie. Chaque code secret, associé au niveau $i$, est codé sur 4 bits.

1. Imaginer un circuit combinatoire qui simule ce jeu. On utilisera pour cela différents circuits connus : multiplexeurs et comparateurs. Enrichir votre circuit avec d’autres circuits combinatoires afin d’indiquer le nombre de niveaux franchis correctement.
2. Sachant que l’on peut envoyer 1 million de codes par seconde, combien de temps faudrait-il, au pire cas, pour tester toutes les combinaisons possibles, dans le cas où le nombre de niveaux est 9 ? Refaites le calcul pour des codes sur 5 bits.

*Cet exercice ne nécessite pas de formules complexes, mais une réflexion architecturale.*

1. Le jeu est inspiré du jeu "Dix de Chute" des années 70.