Architecture : les portes logiques

Du transistor à l'additionneur

> 🎯 **Objectifs de ce chapitre**
>
> - Expliquer comment un transistor, utilisé comme interrupteur, permet de réaliser des portes logiques.
> - Connaître les portes NON, ET, OU, OU exclusif, NAND et NOR : noms (français et anglais), symboles, tables de vérité.
> - Dresser la table de vérité d'un circuit et en déduire l'expression booléenne qu'il réalise.
> - Comprendre qu'un circuit combinatoire réalise une fonction booléenne, et que l'UAL d'un processeur est construite ainsi.

I. Un peu d'histoire

Les premiers ordinateurs électroniques, comme l'**ENIAC** (1945-1946), fonctionnaient avec des **tubes à vide**. L'ENIAC en comptait près de 17 500, pesait environ 30 tonnes et consommait environ 150 kW. Les tubes chauffaient beaucoup et tombaient régulièrement en panne. Il n'était pas programmé par une mémoire, mais par des câbles et des commutateurs (voir le cours sur l'architecture de von Neumann).

Le **transistor**, inventé en 1947, est plus petit, plus fiable et consomme beaucoup moins : à partir des années 1950, il remplace les tubes. On apprend ensuite à graver plusieurs composants sur une même plaquette de matériau semi-conducteur : c'est le **circuit intégré**, dont le premier date de 1958. Depuis, le nombre de transistors par puce a explosé.



Figure 1 — Un tube électronique

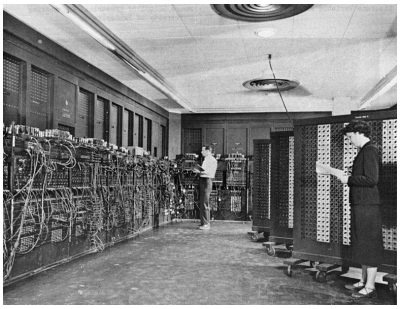

Figure 2 — L'ENIAC

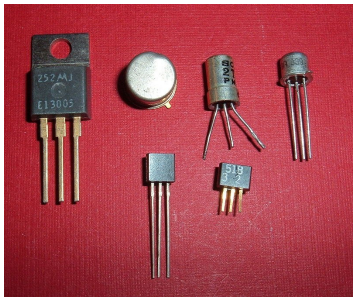

Figure 3 — Quelques transistors

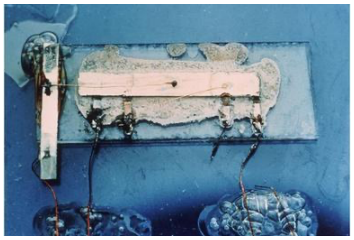

Figure 4 — Le premier circuit intégré (1958, environ 11 × 1,6 mm)

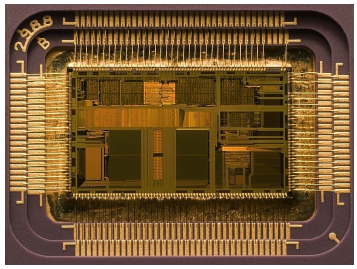

Figure 5 — L'intérieur d'un Intel 80486DX2 (plus d'un million de transistors)

Année	Événement	Ordre de grandeur
1946	ENIAC : programmé par câblage	17 500 tubes à vide
1947	Invention du transistor (laboratoires Bell)	1 transistor
1958	Premier circuit intégré (Jack Kilby, Texas Instruments)	quelques composants
1971	Intel 4004, premier microprocesseur	2 300 transistors
1989	Intel 80486	environ 1,2 million de transistors
2024	Puce Apple M4	28 milliards de transistors

La finesse de gravure d'un transistor est passée d'environ 1 µm (soit 1 000 nm) pour le 80486 à quelques nanomètres aujourd'hui (le « 3 nm » de l'Apple M4 est surtout un nom commercial).

> 🔬 **Essaie, observe — combien de doublements ?**
>
> Entre l'Intel 4004 (1971, 2 300 transistors) et l'Apple M4 (2024, 28 milliards de transistors) :
>
> 1. Par combien le nombre de transistors a-t-il été multiplié ? .....
>
> 2. Combien de fois a-t-il fallu le **doubler** pour obtenir ce facteur ? (aide : $2^{23} = 8\,388\,608$ et $2^{24} = 16\,777\,216$) .....
>
> 3. Sur 53 ans, tous les combien de temps le nombre de transistors a-t-il doublé en moyenne ? .....

> 📘 **La loi de Moore**
>
> Gordon Moore, cofondateur d'Intel, a observé en 1965 puis en 1975 que le nombre de transistors d'une puce **double environ tous les deux ans**. Cette tendance s'est vérifiée pendant plus de cinquante ans : c'est elle qui a rendu les ordinateurs toujours plus puissants.

II. Du transistor à la porte logique

1. Le transistor, un interrupteur commandé

Dans un ordinateur, l'information est représentée par deux niveaux de tension : une tension **haute** (par exemple 5 V pour les anciens composants, environ 1 V aujourd'hui) code **1**, une tension **basse** (proche de 0 V, la *masse*) code **0**. Le composant de base qui manipule ces tensions est le **transistor**.

> 📖 **Définition — le transistor (modèle simplifié)**
>
> Un transistor possède trois bornes : le **collecteur** C, l'**émetteur** E et la **base** B. Il se comporte comme un **interrupteur commandé par la base** : si la base reçoit une tension haute (1), l'interrupteur est **fermé** et le courant passe de C vers E ; sinon (0), il est **ouvert**.

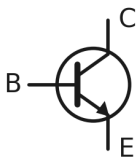

Figure 6 — Symbole d'un transistor (bipolaire NPN)

> ⚠️ **Un modèle simplifié**
>
> Ce modèle est celui d'un transistor bipolaire, vu comme un simple interrupteur. Les puces actuelles utilisent des transistors de type MOS, assemblés selon la technologie CMOS, mais le principe de l'interrupteur commandé reste le même.

2. La porte NON avec un transistor

Voici un montage à un seul transistor. La sortie S est reliée à l'alimentation (niveau 1) par une résistance, et à la masse par le transistor : le collecteur est au point S et l'émetteur à la masse.

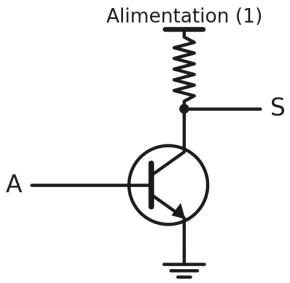

Figure 7 — Un transistor et une résistance

> 🔬 **Essaie, observe**
>
> Pour chaque valeur de A, indique si l'interrupteur est ouvert ou fermé, puis donne la valeur de S (si l'interrupteur est fermé, S est relié à la masse ; sinon S est relié à l'alimentation par la résistance).

A	Transistor (ouvert / fermé)	S
0		
1		

Quelle porte logique ce montage réalise-t-il ? .....

3. La porte NAND avec deux transistors

Plaçons maintenant deux transistors **en série**, l'un à la suite de l'autre : le courant ne peut aller vers la masse que si **les deux** interrupteurs sont fermés.

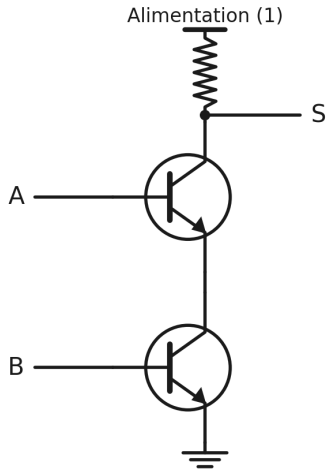

Figure 8 — Deux transistors en série

🔬 Essaie, observe

A	B	Transistor 1	Transistor 2	Courant vers la masse ?	S
0	0				
0	1				
1	0				
1	1				

S vaut 0 dans un seul cas. Lequel ? Quelle porte logique obtient-on ? .....

📘 À retenir

- Un transistor est un **interrupteur commandé** par sa base.
- Avec un, deux ou quelques transistors, on réalise des **portes logiques** : un transistor donne un NON, deux transistors en série un NAND (en parallèle, un NOR : exercice 1 du TD).
- Une puce contient des milliards de transistors assemblés en portes. **À partir de maintenant, on oublie les transistors** et on raisonne uniquement avec des portes.

💡 Pour aller plus loin — nandgame

Le jeu nandgame.com fait construire un ordinateur à partir d'un seul composant. D'après sa description, son premier niveau consiste à construire une porte NAND à partir de composants de base. Il se joue dans un navigateur (s'il n'est pas bloqué par le filtre de l'établissement).

III. Les portes logiques

> 📖 **Définition — porte logique et table de vérité**
>
> Une **porte logique** est un circuit qui réalise une opération booléenne : elle prend un ou plusieurs **bits** en entrée et fournit un bit en sortie. Sa **table de vérité** donne la sortie pour chaque combinaison des entrées : avec n entrées, elle a 2^n lignes.

Vous connaissez déjà les opérateurs `not`, `and` et `or` de Python : ce sont les opérations réalisées par les portes NON, ET et OU.

1. Les portes de base

Compléter les tables de vérité avec votre professeur.

Porte	Symbole	A	B	S
NON NOT		0	—	
		1	—	
ET AND		0	0	
		0	1	
		1	0	
		1	1	
OU OR		0	0	
		0	1	
		1	0	
		1	1	
OU exclusif XOR		0	0	
		0	1	
		1	0	
		1	1	

> 📖 **OU inclusif et OU exclusif**
>
> En logique, le OU est **inclusif** : A OU B vaut 1 si A vaut 1, si B vaut 1, ou si les deux valent 1. Dans « du lait ou du sucre dans votre café ? », on peut prendre les deux : c'est un OU inclusif.
>
> Le **OU exclusif** (XOR) vaut 1 si **un seul** des deux bits vaut 1. Dans « fromage ou dessert », on ne peut pas prendre les deux : c'est un OU exclusif.

2. Les portes avec inversion

On obtient trois autres portes en ajoutant un NON à la sortie d'une porte ET, OU, ou OU exclusif. Un petit rond sur le symbole indique l'inversion.

Porte	Symbole	A	B	S
NON ET NAND		0	0	1
		0	1	1
		1	0	1
		1	1	0
NON OU NOR		0	0	1
		0	1	0
		1	0	0
		1	1	0
NON OU exclusif XNOR		0	0	1
		0	1	0
		1	0	0
		1	1	1

> 🔬 **Essaie, observe**
>
> Comparer la table de chaque porte avec inversion à celle de la porte correspondante (ET, OU, OU exclusif). Que remarquez-vous ? .....
>
> La porte XNOR vaut 1 quand ses deux entrées sont ..... : on l'appelle aussi la porte d'équivalence.

IV. Les circuits combinatoires

1. Qu'est-ce qu'un circuit combinatoire ?

> 📖 **Définition — circuit combinatoire**
>
> En reliant des portes logiques, on obtient un **circuit**. Un circuit est **combinatoire** si ses sorties ne dépendent que de la valeur **actuelle** de ses entrées : il n'a pas de mémoire. Il réalise une **fonction booléenne**.

Un circuit combinatoire peut se décrire de trois façons équivalentes : par son **schéma**, par son **expression booléenne** et par sa **table de vérité**. Voici un exemple, le **multiplexeur** : il recopie en sortie l'entrée A si `Sel` vaut 0, et l'entrée B si `Sel` vaut 1.

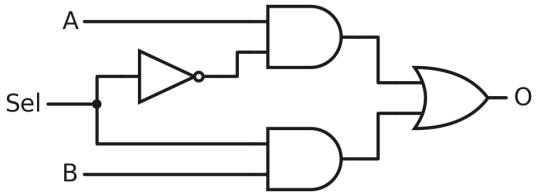

Figure 9 — Un multiplexeur

Son expression booléenne est : **O = (A ET NON Sel) OU (B ET Sel)**.

2. Étudier un circuit

> 📘 **Méthode — dresser la table de vérité d'un circuit**
>
> - Lister les **entrées** et écrire les 2^n combinaisons dans l'ordre **binaire** : A est le bit de poids fort, donc 000, 001, 010, 011, 100, 101, 110, 111 pour trois entrées.
> - Nommer la **sortie de chaque porte**, en allant des entrées vers la sortie du circuit.
> - Remplir la table **colonne par colonne**, dans cet ordre.
> - Lire la dernière colonne : c'est la fonction réalisée par le circuit.

🔬 Essaie, observe — la table du multiplexeur

Compléter la table du multiplexeur de la figure 9 avec votre professeur.

A	B	Sel	NON Sel	A ET NON Sel	B ET Sel	O
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

Que vaut O quand Sel = 0 ? Quand Sel = 1 ? .....

3. Deux circuits, une même fonction

Des circuits différents peuvent avoir la **même table de vérité** : ils réalisent alors la même fonction. On écrit une **égalité entre expressions booléennes**. Par exemple, `NON(NON(A) OU NON(B))` et `A ET B` ont la même table : les exercices 2 et 3 du TD en donnent deux exemples.

💡 Pour aller plus loin — les lois de De Morgan

Les deux égalités `NON(A ET B) = NON(A) OU NON(B)` et `NON(A OU B) = NON(A) ET NON(B)` s'appellent les **lois de De Morgan**. Elles permettent de transformer un ET en OU (et inversement) à l'aide de portes NON.

4. Le NAND, une porte universelle

📘 Propriété — le NAND suffit

On peut réaliser **toutes** les portes logiques en n'utilisant que des portes NAND (ou que des portes NOR).

Porte	Réalisée avec des NAND
NON A	NAND(A, A)
A ET B	NAND(NAND(A, B), NAND(A, B))
A OU B	NAND(NAND(A, A), NAND(B, B))

Essaie, observe

Vérifie la première ligne du tableau : complète la table de vérité de NAND(A, A) et compare-la à celle de NON A.

A	NAND(A, A)	NON A
0		
1		

Que remarques-tu ? .....

Pour aller plus loin — l'ordinateur d'Apollo

L'ordinateur de bord des missions lunaires Apollo (années 1960) était construit avec environ 5 600 portes NOR, réparties dans des circuits intégrés. Pour en savoir plus : righto.com — a computer built from NOR gates.

Et les mémoires ?

Les mémoires d'un ordinateur sont elles aussi réalisées avec des portes, mais elles ne sont **pas combinatoires** : leur sortie dépend de ce qui a été mémorisé auparavant. Ce n'est pas au programme.

V. Additionner avec des portes

1. Additionner deux bits : le demi-additionneur

Quand on additionne deux bits A et B, le résultat vaut 0, 1 ou 2 (soit 10 en binaire). Il faut donc **deux sorties** : la **somme** S (le bit de droite) et la **retenue** R (le bit de gauche).

Essaie, observe

Compléter la table avec votre professeur, puis chercher une porte logique pour S et une pour R.

A	B	A + B (en binaire)	S	R
0	0	00		
0	1	01		
1	0	01		
1	1	10		

S est la sortie de la porte ..... ; R est la sortie de la porte .....

> 📖 **Définition — demi-additionneur**
>
> Un **demi-additionneur** additionne deux bits A et B. Il fournit la somme **S = A XOR B** et la retenue **R = A ET B**.

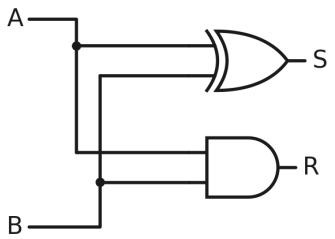

Figure 10 — Un demi-additionneur

2. Additionner trois bits : l'additionneur complet

Pour additionner des nombres de plusieurs bits, il faut tenir compte de la retenue de la colonne précédente, comme dans une addition posée à la main. On additionne donc **trois bits** : A, B et la retenue d'entrée **Re**. Le résultat vaut de 0 à 3 : on obtient la somme **S** et la retenue de sortie **Rs**.

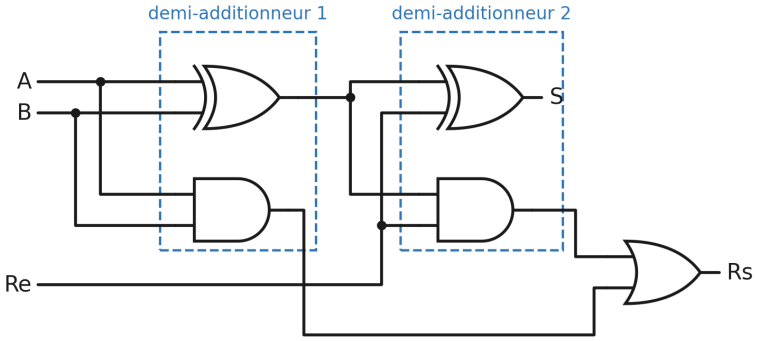

Figure 11 — Un additionneur complet : deux demi-additionneurs et une porte OU

Essaie, observe — la table de l'additionneur complet

A	B	Re	A + B + Re (en binaire)	S	Rs
0	0	0	00		
0	0	1	01		
0	1	0	01		
0	1	1	10		
1	0	0	01		
1	0	1	10		
1	1	0	10		
1	1	1	11		

Compare avec le demi-additionneur : que se passe-t-il quand Re = 0 ? .....

3. Additionner des nombres de plusieurs bits

En enchaînant des additionneurs complets, la retenue de sortie de chacun devient la retenue d'entrée du suivant. Le premier reçoit Re = 0. Avec quatre additionneurs, on additionne deux nombres de 4 bits :

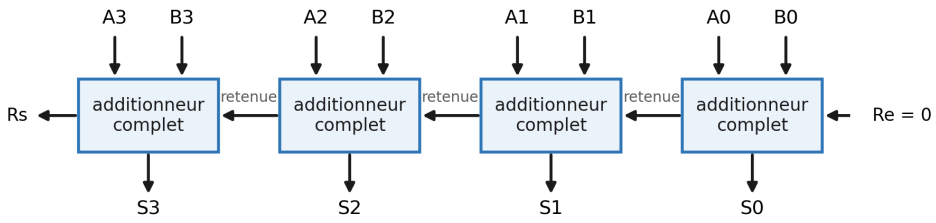

Figure 12 — Additionneur sur 4 bits

Essaie, observe — 5 + 7

Additionner A = 0101 (5) et B = 0111 (7) en suivant la retenue de droite à gauche. Complète le tableau, en recopiant chaque fois Rs dans la ligne suivante, à la colonne Re :

Rang	A	B	Re	S	Rs
bit 0	1	1	0		
bit 1	0	1			
bit 2	1	1			
bit 3	0	0			

Le résultat S3 S2 S1 S0 est ..... , soit ..... en décimal.

📘 **Le lien avec le processeur**

L'**UAL** d'un processeur contient de tels circuits : des additionneurs, et d'autres portes pour les opérations logiques. C'est elle qui exécute une instruction comme `ADD` du cours sur l'architecture de von Neumann.

À retenir

📘 **L'essentiel**

- Un **transistor** est un interrupteur commandé ; quelques transistors forment une **porte logique**.
- Les portes de base sont **NON**, **ET**, **OU**, **OU exclusif** ; avec inversion : **NAND**, **NOR**, **XNOR**. Chacune est décrite par sa **table de vérité** (2^n lignes pour n entrées).
- Un **circuit combinatoire** a des sorties qui ne dépendent que des entrées : il réalise une **fonction booléenne**, décrite par un schéma, une expression ou une table.
- Le **NAND** (ou le NOR) suffit pour réaliser toutes les portes.
- Un **demi-additionneur** (S = A XOR B, R = A ET B) puis un **additionneur complet** permettent d'additionner des nombres binaires : c'est le cœur de l'**UAL**.

✏️ **Auto-test**

a. Combien de lignes a la table de vérité d'un circuit à 4 entrées ? .....

b. Écris NON(A) en n'utilisant que des portes NAND : .....

c. Pour A = 1 et B = 1, que valent S et R dans un demi-additionneur ? Pourquoi ? .....

💡 **Pour continuer**

- Le **TD sur les portes logiques** (fichier séparé) : exercices sur les circuits, avec corrigés.
- Le **simulateur de circuits de Capytale** : pour construire et tester un demi-additionneur puis un additionneur complet.
- Le jeu nandgame.com (si l'établissement ne le bloque pas).

Crédits des illustrations

Figure	Source	Licence
1	Wikipédia (tube électronique)	Domaine public
2	Wikipédia (ENIAC)	Domaine public
3	Wikipédia (transistors)	CC BY-SA 3.0 — Arnold Reinhold
4	Circuit intégré de J. Kilby, 1958	Courtesy of Texas Instruments © Texas Instruments Object ID 500003095
5	Wikipédia (Intel 80486DX2)	C C BY SA 3.0 Matt Britt
6 à 12	Schémas réalisés pour ce cours	—

Licence CC BY-SA 3.0 : http://creativecommons.org/licenses/by-sa/3.0/. Le nom de l'auteur de chaque photo doit être indiqué sur la page Wikipédia de l'image.