Math93.com

Cours - NSI
Circuits et logique booléenne

Prérequis et Thème

— Prérequis : variables booléennes.

— Notions introduites : transistors, portes logiques / tables de vérité / fonctions et expressions booléennes / circuits combinatoires.

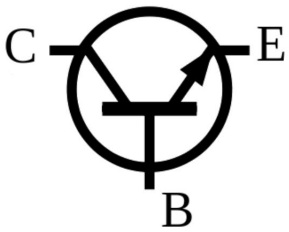
1 Portes logiques

Les circuits d'un ordinateur (mémoire, microprocesseur, ...) manipules uniquement des chiffres binaires 0 et 1 qui sont simplement représentés par des tensions électriques. Le chiffre 0 est associée à une tension basse et le 1 à une haute.

1.1 Les transistors

Les transistors, de l'anglais *transfer resistor* (résistance de transfert), que l'on trouve dans les circuits se comportent comme des interrupteurs. De nos jours, le transistor est tellement miniaturisé que votre ordinateur ou votre téléphone en contient plusieurs milliards sur une simple puce de silicium de la taille de votre ongle!

Symbole du transistor



1.1.1 Fonctionnement

Le transistor (du type CMOS) comporte trois connexions externes :

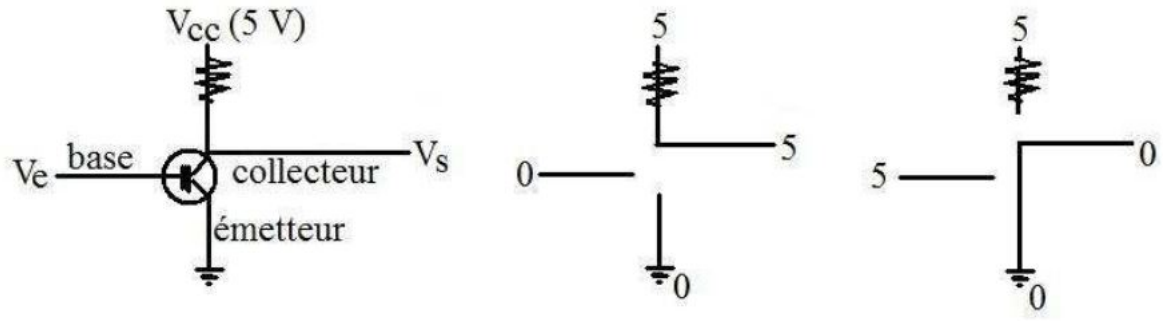
— Le collecteur (C) est la sortie du transistor, et il est relié au fil d'où vient la tension Vcc (5 volts de courant continu) de l'alimentation.

— L'émetteur (E) est relié à la masse (0 volt).

— La base (B) constitue la connexion d'entrée. Tout dépend de la tension Ve qui lui est appliquée.

  — Si l'on n'applique aucune tension à la base (Ve = 0), le transistor bloque le courant entre collecteur et émetteur, et la sortie passe à la tension Vs = 5 volts.

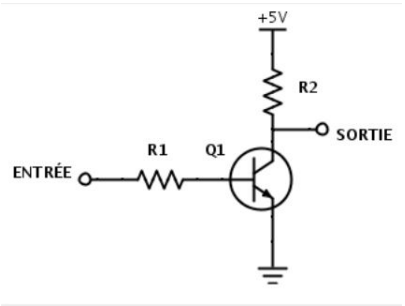
  — Si l'on met sur la base une tension de Ve = 5 volts en entrée, le courant passe entre le collecteur et l'émetteur, ce qui met la sortie à la masse, soit Vs = 0 volt.



Le transistor inverseur, avec blocage ou passage selon la valeur binaire de la tension Ve.
Ainsi le niveau haut de l'entrée donne un niveau bas à la sortie, et vice versa. En termes logiques, 0 est transformé en 1, et 1 en 0. On obtient un inverseur, correspondant à la porte logique baptisée NON

1.2 Les portes logiques et fonctions associées

1.2.1 Porte NON (NOT) : fonction non, négation logique



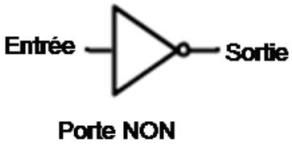


Table de vérité : loi NOT

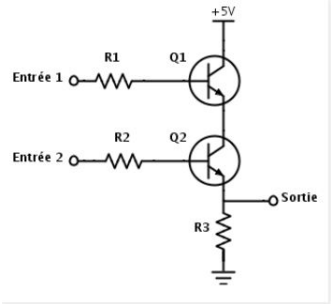
Entrée	Sortie
a	$\overline{a} = non(a) = \neg(a)$
0	1
1	0

Autres notations : $not(a)$, $\backsim a$

La fonction booléenne de passage s'écrit $\boxed{f(a) = \overline{a}}$, où a désigne l'entrée et $f(a)$ la sortie, étant entendu est par définition le contraire (ou le complément) de a.

1.2.2 Porte ET (AND) : fonction ET, Conjonction logique => ×

La porte ET peut être réalisée au moyen de deux transistors : pour que la sortie soit à 5 V, il faut que les deux jonctions collecteur-émetteur soient conductrices, donc que les deux entrées soient à 5 V. Aussitôt qu'une entrée se trouve à 0 V, le transistor qui lui est associé empêche la circulation du courant dans la résistance R3, donc la sortie sera à 0 V.



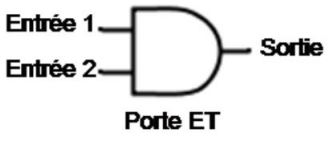


Table de vérité : loi AND

Entrées		Sortie
a	b	$a.b = a \wedge b$
0	0	0
0	1	0
1	0	0
1	1	1

Autres notations : a&b

La fonction logique ET est définie de la manière suivante :

a ET b est VRAI si et seulement si a est VRAI et b est VRAI.

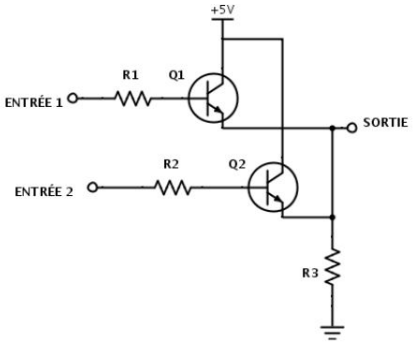
Cette loi est aussi notée :

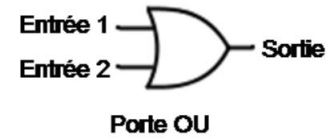
. ; $\wedge$; & ; && (en Perl, C, PHP) ; AND (en Python, Pascal, Ada, PHP)

La fonction booléenne de passage s'écrit $\boxed{f(a,b) = a.b = ab = a \wedge b}$.

1.2.3 Porte OU (OR) : fonction OU, disjonction logique => +

La porte OU peut être réalisée au moyen de deux transistors : pour qu'un courant circule dans la résistance R3, il s'agit qu'un ou l'autre des deux transistors le laisse circuler, ce qui sera le cas si une ou l'autre des entrées est à 5 V afin qu'un courant circule dans la base.





Porte OU

Table de vérité : loi OU

Entrées		Sortie
a	b	$a+b=a\vee b$
0	0	0
0	1	1
1	0	1
1	1	1

Autres notations : $a \mid b$, a or b , a OR b

La fonction logique OU est définie de la manière suivante :

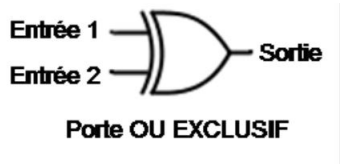
a OU b est VRAI si et seulement si a est VRAI OU b est VRAI.

La fonction booléenne de passage s'écrit :

$$\boxed{f(a,b)=a+b=a\vee b}$$

1.2.4 Porte OU exclusif (XOR) : fonction XOR , disjonction exclusive => $\oplus$

La sortie d'une porte "OU EXCLUSIF" prend l'était logique "HAUT" si une seule de ses entrées est à l'état logique "HAUT", mais pas l'autre. En d'autres mots, la sortie sera basse si les deux entrées sont dans le même état logique, et la sortie sera haute si les deux entrées sont des des états différents.



Porte OU EXCLUSIF

Table de vérité : loi XOR

Entrées		Sortie
a	b	$a\oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

La fonction logique OU eXclusif (XOR) est définie de la manière suivante :

a XOR b est VRAI si et seulement si a est VRAI OU b est VRAI mais pas les deux.

La fonction booléenne de passage s'écrit :

$$\boxed{f(a,b)=a\bigoplus b}$$

1.2.5 Les portes négations : NAND, NOR et XNOR

On peut obtenir des portes par négation des portes AND, OR et XOR.

1. **Porte NAND, NON ET, NON AND**

Porte NON-ET

$$f(a,b) = \overline{a.b}$$

Table de vérité : loi NAND

Entrées		Sortie AND	Sortie NAND
a	b	$a.b$	$\overline{a.b}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

2. **Porte NOR, NON OU, NON OR**

Porte NON-OU

$$f(a,b) = \overline{a+b}$$

Table de vérité : loi NOR

Entrées		Sortie OR	Sortie NOR
a	b	$a+b$	$\overline{a+b}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

3. **Porte XNOR, NON XOR**

Porte NON-OU EXCLUSIF

$$f(a,b) = \overline{a \oplus b}$$

Table de vérité : loi XNOR

Entrées		Sortie XOR	Sortie NXOR
a	b	$a \oplus b$	$\overline{a \oplus b}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

2 Fonctions booléennes

Certains circuits électroniques peuvent être décrits comme des fonctions booléennes, c'est à dire des fonctions qui prennent en entrées un ou plusieurs bits et en sortie un unique bit.
On en a rencontré plusieurs, par exemple les fonctions booléennes associées aux portes **NON, ET, OU** (on verra qu'elles forment une base complète) ou celles associées aux portes **NON ET (NAND), NON OU (NOR)** (qui forment aussi une base complète).

2.1 Tables de vérité

On peut définir une fonction booléenne f par sa table de vérité.
Par exemple la fonction booléenne f définie par :

Table de vérité : loi de f

Entrées			Sortie
a	b	c	$f(a,b,c)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Tables de vérité et fonction booléenne

— La table de vérité d'une fonction booléenne à n bits (variables) en entrée aura 2^n lignes.

— On peut montrer qu'il y a 2^{2^n} fonctions logiques booléennes à n variables (bits).

2.2 Fonctions booléennes et base

Fonction booléenne et base

— Les fonctions booléennes élémentaires associées aux portes **NAND** et **NOR** forment une base complète pour la construction des autres fonctions booléennes. C'est à dire que toute fonction booléenne peut être définie à l'aide de ces deux fonctions.

— Il en est de même avec les trois fonctions associées aux portes **NON, ET, OU**.

Pour simplifier la définition des fonctions booléennes, on utilisera plutôt les fonctions élémentaires de base.

Exemple

Par exemple en écrivant les tables de vérité on peut montrer que :

$$a \oplus b = a.\overline{b} + \overline{a}.b$$

Remarque historique

Les opérateurs booléens portent le nom du mathématicien Georges Boole qui inventa au 19e siècle ce calcul. On parle d'algèbre de Boole

2.3 Expressions booléenne

En utilisant ces opérateurs, on peut définir n'importe qu'elle fonction booléenne. Par exemple on peut définir la fonction booléenne f par :

$$f(a,b) = (a.b) \bigoplus (\overline{a} + b)$$

Expression que l'on peut écrire :

$$(a \text{ AND } b) \text{ XOR } \Big(\text{ (NON a) OR b } \Big)$$

2.4 Opérations élémentaires et Lois de Morgan

On peut facilement montrer en complétant les tables de vérités plusieurs égalités sur les opérations. Pensez que le bit 1 est associée à Vrai et 0 à Faux.

Lois de Morgan	$\overline{a+b} = \overline{a}.\overline{b}$	$\overline{a.b} = \overline{a} + \overline{b}$
Involutif	$\overline{\overline{a}} = a$	
Neutre	$1.a = a$	$0 + a = a$
Absorbant	$0.a = 0$	$1 + a = 1$
Idempotence	$a.a = a$	$1 + a = a$
Complément	$a.\overline{a} = 0$	$a + \overline{a} = 1$
Commutativité	$a.b = b.a$	$a + b = b + a$
Associativité	$a.(b.c) = (a.b).c$	$a + (b + c) = (a + b) + c$
Distributivité	$a.(b + c) = (a.b) + (a.c)$	$a + (b.c) = (a + b).(a + c)$

Remarque historique

Les lois de Morgan ont été formulées par le mathématicien britannique Augustus De Morgan (1806-1871).