III

Interrupteurs commandés et portes logiques

Plan du chapitre

I Interrupteurs commandés par une tension

I.1 Premières idées : synthèse de fonctions logiques par association d'interrupteurs

Considérons un lampadaire équipé d'un interrupteur et branché sur une prise murale elle-même commandé par un interrupteur. le lampadaire ne pourra évidemment être allumé que si les deux interrupteurs sont fermés. Comme seuls deux états sont possibles par interrupteur ("ouvert" ou "fermé"), cette association aura 4 états possibles :

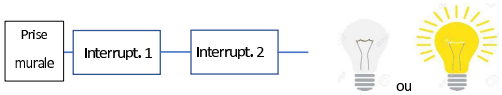


FIGURE III.1 – Lampadaire commandé par deux interrupteurs

Interrupt. 1	Interrupt. 1	Etat lampadaire
ouvert	ouvert	éteint
fermé	ouvert	éteint
ouvert	fermé	éteint
fermé	fermé	allumé

On constate que cette association dite "série" réalise une opération de logique booléenne appelée **"fonction ET"** (AND) puisque le lampadaire n'est allumé que si l'interrupteur 1 et l'interrupteur 2 sont tous les deux fermés.

Une autre possibilité d'association est de placer 2 interrupteurs en parallèle. Ce dispositif est par exemple employé dans la commande de lève-vitre d'un véhicule : la vitre côté passager avant est commandable non seulement par l'interrupteur du passager mais également par un interrupteur dédié côté conducteur : un appui sur l'un ou l'autre actionnera la vitre. Là-encore, il y a 4 états possibles :

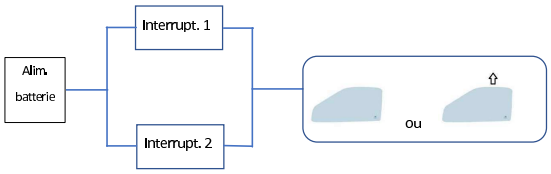


FIGURE III.2 – Lève-vitre commandé par deux interrupteurs

Interrupt. 1	Interrupt. 1	Etat lève-vitre
ouvert	ouvert	pas de mouvement
fermé	ouvert	mouvement
ouvert	fermé	mouvement
fermé	fermé	mouvement

Cette nouvelle association, dite "parallèle", réalise une opération de logique booléenne appelée **"fonction OU"** (OR) puisque la vitre est actionné si l'un ou l'autre des deux interrupteurs (ou bien les deux sont fermés).

IDÉE : on va utiliser des interrupteurs dits "commandés" par des tensions pour réaliser des opérations de logique booléenne.

I.2 Interrupteurs commandés par tension : transistors MOS

Les transistors peuvent, sous conditions, fonctionner comme des interrupteurs commandés ; les transistors employés aujourd'hui dans les circuits logiques sont de type FET (pour **F**ield **E**ffect **T**ransistor ou TEC en français : **T**ransistor à **E**ffet de **C**hamp) ; ils sont au nombre de deux : transistor à canal N et transistor à canal P, et font appel à la technologie MOS pour `Metal oxyde semi-conductor`. Ils comportent, comme tous les transistors, 3 connexions appelées `Grille G` , `Drain D`, et `Source S`. Les circuits intégrés modernes sont constitués de ces deux type transistors complémentaires et sont dits CMOS pour `Complémentary MOS`.

SYMBOLES :

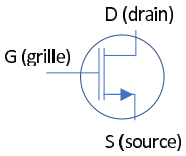


FIGURE III.3 – TEC canal N

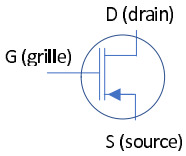


FIGURE III.4 – TEC canal P

Le comportement de ces transistors dépend du potentiel appliqué sur la grille :

	$V_G = 0\ V$	$V_G = V_{cc} > 0$
transistor N	**DS ouvert**	**DS fermé**
transistor P	**DS fermé**	**DS ouvert**

II Associations d'interrupteurs commandés - portes logiques

II.1 Quelques définitions et relations essentielles

Les interrupteurs commandés par tension sont aujourd'hui à la base du fonctionnement des circuits numériques modernes. Ces dispositifs dits "logiques" s'appuient sur l'algèbre de Boole pour laquelle il existe simplement deux valeurs possibles des variables logiques que l'on notera $\{a, b \ldots\}$:

$$B = \{Vrai, Faux\} \qquad ou \qquad B = \{0, 1\}$$

En outre, **seulement 3 opérations principales** sont à la base de cette algèbre (toutes les autres opérations se composant à partir de celles-ci) : LA NÉGATION, LA CONJONCTION, et LA DISJONCTION.

Définition II-1: NÉGATION

«la négation de a est `VRAI` (1) ssi a est faux».

Notation : $\boxed{NOT\ a = \overline{a}}$

CONSÉQUENCES : $\overline{0} = 1$ et $\overline{1} = 0$. On peut résumer les propriétés de la conjonction dans une **table de vérité ou table de loi** :

Négation	
a	$\overline{a}$
0	1
1	0

Définition II-2: CONJONCTION

«(a ET b) est VRAI (1) ssi a est vrai et b est vrai».

Notation : $\boxed{a \cdot b = a \wedge b = a\&b}$

CONSÉQUENCES : on a la la table de vérité suivante

Conjonction		
a	b	$a \cdot b$
0	0	0
1	0	0
0	1	0
1	1	1

Définition II-3: DISJONCTION

«(a OU b) est VRAI (1) ssi a est vrai ou b est vrai».

Notation : $\boxed{a + b = a \vee b = a|b}$

CONSÉQUENCES : la table de vérité est alors :

Disjonction		
a	b	$a + b$
0	0	0
1	0	1
0	1	1
1	1	1

QUELQUES PROPRIÉTÉS IMPORTANTES :

Pour 3 variables logiques $\{a, b, c\}$, on a :

- ASSOCIATIVITÉ : $(a + b) + c = a + (b + c) = a + b + c$
- COMMUTATIVITÉ : $a + b = b + a$ et $a \cdot b = b \cdot a$
- DISTRIBUTIVITÉ : $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$ et $a + (b \cdot c) = (a + b) \cdot (a + c)$
- LOIS DE DE MORGAN : $\overline{a + b} = \overline{a} \cdot \overline{b}$ et $\overline{a \cdot b} = \overline{a} + \overline{b}$

Exercice de cours: **(II.1) - n° 1.** *Démontrer les lois de De Morgan à l'aide de tables de vérité.*

II.2 Premier exemple d'association : la porte logique NON (NOT) ou inverseur

Définition II-4: PORTE LOGIQUE

Une porte logique est un circuit numérique réalisant une opération booléenne entre des variables logiques représentées par des tensions appliquées sur les entrées de la porte $\{e_1, e_2, \ \ldots \ e_n\}$; le résultat de l'opération est renvoyé sur la sortie s de la porte et est également représenté par une tension.
Le variables logiques ne pouvant prendre que deux valeurs : 0 (état bas) et 1 (état haut), il en est de même avec les tensions les représentant ; par exemple pour les circuits dits TTL, on utilise les valeurs $u_{bas} = 0\ V$ pour "0", et $u_{haut_{max}} = 5\ V$ pour "1".

On considère l'association d'interrupteurs commandés suivante :

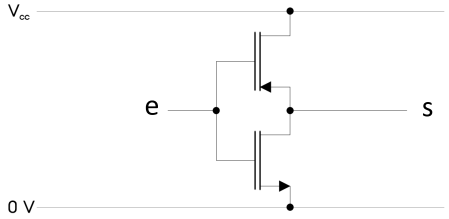


FIGURE III.5 – Assemblage porte NON

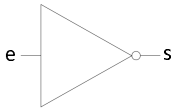
ANALYSE DE COMPORTEMENT :

- si $e = 1$ (i.e. 5V) alors le transistor supérieur (P) est bloquant donc l'interrupteur supérieur est **ouvert** ; en revanche le transistor inférieur (N) est passant, donc l'interrupteur est fermé : la sortie S est donc reliée à la masse.
- si $e = 0$ (i.e. 0V) alors la situation est inversée : la sortie S est donc reliée à $V_{cc} = 5\ V$, donc $s = 1$.

On en déduit **la table de vérité** de cette association qui traduit les états de sortie possibles s en fonction de l'état de l'entrée ss :

e	s
0	1
1	0

Cette association réalise donc la fonction **NON** (NOT) ; son symbole est :



III Les autres portes logiques

III.1 Porte ET-NON (NAND)

On considère cette nouvelle association d'interrupteurs commandés :

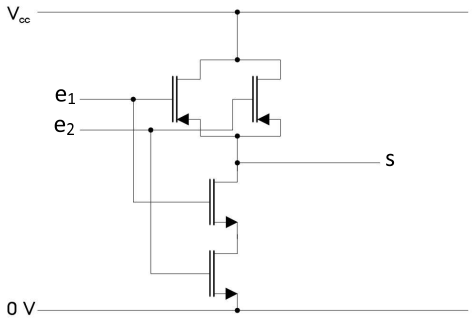


FIGURE III.6 – Assemblage porte NON ET

ANALYSE DE COMPORTEMENT :

- si $e_1 = 1$ et $e_2 = 0$ alors T_1 ouvert mais T_2 fermé donc la sortie est reliée à V_{cc} donc $s = 1$ (et T_3 fermé, T_4 ouvert)
- si $e_1 = 0$ et $\forall e_2$ alors T_1 est fermé donc la sortie est reliée à V_{cc} donc $s = 1$.
- si $e_1 = 1$ et $e_2 = 1$ alors T_3 et T_4 fermés donc la sortie est reliée à $0\ V$ donc $s = 0$.

On en déduit immédiatement la table de vérité :

e_1	e_2	s
0	0	1
1	0	1
0	1	1
1	1	0

qui correspond à la fonction logique **NON ET** dont le symbole est : A INCLURE

En outre, l'opération booléenne correspondant à cette porte est $s = \overline{e_1 \cdot e_2}$ (le surlignage correspond à une inversion de la valeur).

Dans l'hypothèse d'une porte à n entrées, la table de vérité se déduirait immédiatement de la relation :

$$s = \overline{\prod_{i=1}^{n} e_i}$$

Remarque III-1: UNIVERSALITÉ DES PORTES NAND

Toutes les fonctions logiques peuvent être réalisées à l'aide de portes NAND, ce qui est avantageux car elles sont généralement nettement moins coûteuses que les autres. Elles sont assemblées dans des circuits intégrés (par exemple la série des CI7400 à 4 portes NAND)

III.2 Porte ET (AND)

On obtient immédiatement la porte **ET** en ajoutant un inverseur (porte **NOT**) en sortie de l'association d'interrupteurs correspondant à la porte **NAND** décrite ci-dessus :

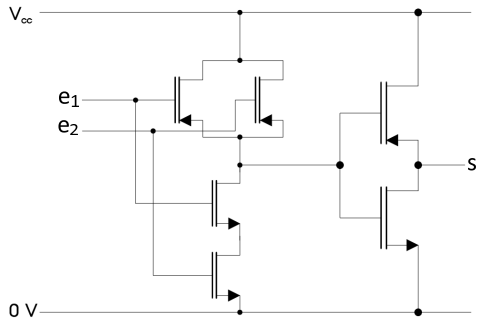


FIGURE III.7 – Assemblage porte ET

La table de vérité de la porte **ET** à deux entrées se déduit immédiatement en prenant pour chaque cas des états d'entrée, la sortie complémentaire obtenue pour la porte **NON ET** :

e_1	e_2	s
0	0	0
1	0	0
0	1	0
1	1	1

L'opération booléenne correspondante est : $s = e_1 \cdot e_2$

Si la porte est à n entrées, l'opération booléenne est :

$$\boxed{s = \prod_{i=1}^{n} e_i}$$

Exercice de cours: **(III.2) - n° 2.** *Construire la fonction AND à partir de portes* NAND.

III.3 Portes OU (OR)

On considère l'association suivante dans laquelle on remarque que le second circuit est un simple inverseur :

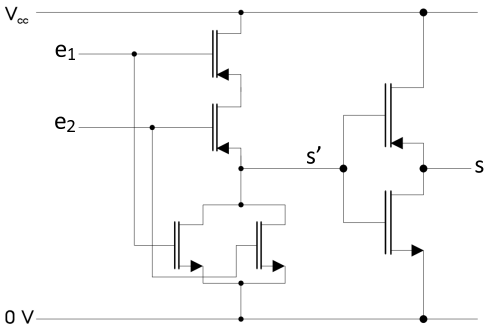


FIGURE III.8 – Assemblage porte OU

ANALYSE DE COMPORTEMENT :

- si $e_1 = 0$ et $e_2 = 0$ alors T_1 et T_2 fermés (T_3 et T_4 par ailleurs ouverts) donc $s' = 1$ donc après inversion $s = 0$.
- si $e_1 = 1$ alors T_1 ouvert et T_3 fermé $\forall e_2$ donc $s' = 0$ donc après inversion $s = 1$.

- si $e_2 = 1$ alors T_2 ouvert et T_4 fermé $\forall e_2$ alors $s' = 0$ donc après inversion $s = 1$.

La table de vérité est donc :

e_1	e_2	s
0	0	0
1	0	1
0	1	1
1	1	1

L'opération booléenne correspondante est donc : $s = e_1 + e_2$

Si la porte est à n entrées, l'opération booléenne correspondante est :

$$\boxed{s = \sum_{i=1}^{n} e_i}$$

Exercice de cours: **(III.3) - n° 3.** *Construire la fonction OU à partir de portes* NAND.

III.4 Porte logique OU-NON (NOR)

Cette porte se déduit immédiatement de la porte `OU` en retirant l'inverseur qui apparait en seconde partie de circuit.

Sa table de vérité est donc :

e_1	e_2	s
0	0	1
1	0	0
0	1	0
1	1	0

L'opération booléenne correspondante est : $s = \overline{e_1 + e_2}$.

Si la porte comporte n entrées, on a alors :

$$\boxed{s} = \overline{\sum_{i=0}^{n} e_i}$$

Exercice de cours: **(III.4) - n° 4.** *Construire la fonction OU NON à partir de portes* NAND.

III.5 Porte logique OU EXCLUSIF (XOR)

On considère l'association d'interrupteurs suivante :

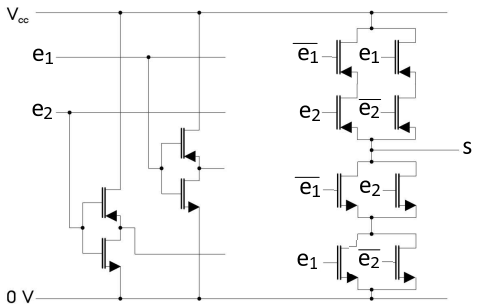


FIGURE III.9 – Assemblage porte OU EXCLUSIF

La table de vérité se déduit facilement par une démarche analogue à celles menées plus haut :

e_1	e_2	s
0	0	0
1	0	1
0	1	1
1	1	0

Remarque III-2: CONSTRUCTION À PARTIR DES AUTRES PORTES

On peut facilement construire la porte XOR à partir des autres portes en réécrivant sa relation logique de définition. D'après la table de vérité, on a en opération booléenne :

$$s = e_1 \oplus e_2 = e_1 \cdot \overline{e_2} + \overline{e_1} \cdot e_2$$

soit :

$$s = e_1 \ XOR \ e_2 = (e_1 \ ET \ \overline{e_2}) \ OU \ (\overline{e_1} \ ET \ e_2)$$

ce qui donne le montage suivant :

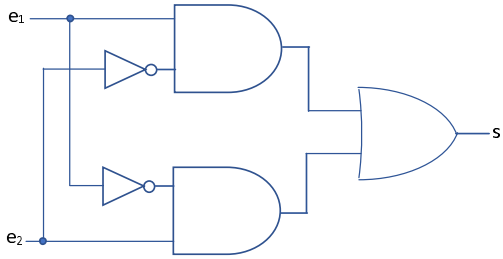


FIGURE III.10 – Assemblage d'une porte **OU EXCLUSIF** à l'aide des portes **NON, ET, et OU**

IV Exemple simple d'application : l'additionneur

Les additionneurs sont des circuits logiques réalisant une opération d'addition ; ils sont notamment très présents dans les unités de calculs logiques des processeurs d'ordinateurs (ALU pour **A**rithmetic-**L**ogic **U**nit).

On se propose de présenter le principe de fonctionnement de tels opérateur sur le cas très simple d'un additionneur 1 bit.

IV.1 1/2 Additionneur (Half Adder ou "HA") 1 BIT

On étudie ici un 1/2 additionneur à 1 Bit (ou HA) qui additionne les deux bits d'entrée, calcule la retenue, mais ne la propage pas au bit suivant ; la table de vérité est la suivante :

e_1	e_2	s	c (carry pour retenue)
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

On constate que le 1/2 additionneur est une porte **XOR** pour l'opération de somme et une porte **AND** pour la retenue. Son schéma logique est donc le suivant :

EXPRESSION LOGIQUE : $\begin{cases} s = e_1 \oplus e_2 = e_1 \cdot \overline{e_2} + \overline{e_1} \cdot e_2 \\ c = e_1 \cdot e_2 \end{cases}$

Exercice de cours: **(IV.1) - n° 5.** *Proposer un schéma de 1/2 additionneur constitué exclusivement de portes* **NAND**. *On pourra utiliser le théorème de De Morgan sous la forme :* $e_1 + e_2 = \overline{\overline{e_1} \cdot \overline{e_2}}$

IV.2 Additionneur complet 1 bit (Full Adder)

IDÉE : pourvoir tenir compte des retenues dans les additions des bits successifs pour faire un additionneur complet.

Pour pouvoir additionner correctement les bits de poids supérieurs e_{1_i} et e_{2_i}, il faut tenir compte de la retenue c_{i-1} propagée depuis le rang $i-1$. Un additionneur complet comporte donc 1 entrée de plus que le 1/2 additionneur pour la retenue en provenance de l'addition des bits de poids inférieur c_{in} et une sortie supplémentaire pour la retenue en direction des bits de poids supérieur c_{out}. Sa table de vérité est :

e_1	e_2	c_{in}	s	c_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

EXPRESSION LOGIQUE :

D'après la table de vérité, on a :

$s = \overline{e_1} \cdot \overline{e_2} \cdot c_{in} + \overline{e_1} \cdot e_2 \cdot \overline{c_{in}} + e_1 \cdot \overline{e_2} \cdot \overline{c_{in}} + e_1 \cdot e_2 \cdot c_{in}$

$= \overline{c_{in}} \cdot (\overline{e_1} \cdot e_2 + e_1 \cdot \overline{e_2}) + c_{in}(\overline{e_1} \cdot \overline{e_2} + e_1 \cdot e_2)$

$= \overline{c_{in}} \cdot (e_1 \oplus e_2) + c_{in} \cdot (\overline{e_1 \oplus e_2})$

$= c_{in}\ XOR\ e_1\ XOR\ e_2$

et :
$c_{out} = c_{in}(\overline{e_1} \cdot e_2) + e_1 \cdot e_2 \cdot (c_{in} + \overline{c_{in}}) = c_{in} \cdot (e_1 \ XOR \ e_2) + e_1 \cdot e_2$

Le schéma d'un **additionneur complet** assemblé à partir des portes classiques est donc :

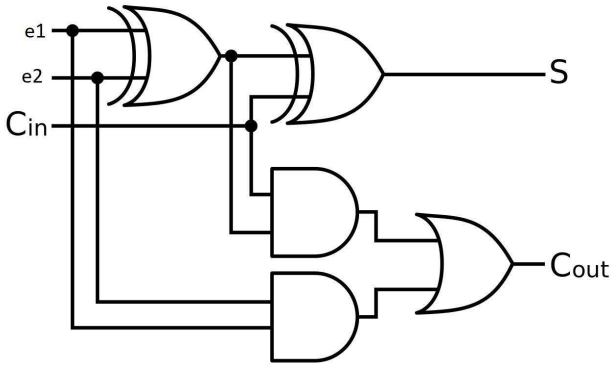


FIGURE III.11 – Additionneur complet 1 bit