Baccalauréat STI2D, classe de première : Enseignement Transversal	
 **Chaîne d'information**	

Sciences et Technologies de l'Industrie et du Développement Durable

4 : La chaîne d'information (Traitement de l'information)

Durée: 3 heures

Objectifs : Algèbre de BOOLE.
Présentation des différents opérateurs logiques (ET, OU, NON).
Les autres opérateurs (Nor, Nand, XOR, XNOR, Inhibition).
Réalisation d'une fonction combinatoire simple.
La logique séquentielle, les bascules, exercice.

Prérequis: Aucun.

Bases théoriques : Aucune.

Outils :

Supports :

Modalités : Les « * » font référence à des définitions en fin de chapitre, Ctrl + click sur le mot renvoi directement à la définition.

Synthèse et validation :

Ressources existantes :

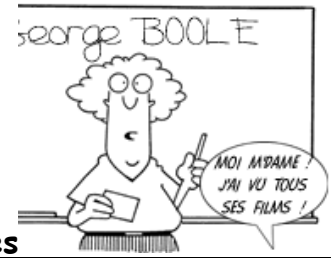
Travail à réaliser : Exercices en fin de chapitre.

Chaîne d'information

Quatrième partie : Traitement de l'information

4.1 – Algèbre de BOOLE

http://www.arcanapercipio.com/lessons/algebre_de_boole/algebre_de_boole.html



4.1.1 - Généralités

L'étude du traitement de l'information nécessite la connaissance de l'algèbre BOOLE.L'**algèbre de Boole**, ou calcul booléen, est la partie des mathématiques, de la logique et de l'électronique qui s'intéresse aux opérations et aux fonctions sur les **Variables logiques***. Plus spécifiquement, l'algèbre booléenne permet d'utiliser des techniques algébriques pour traiter les expressions à deux valeurs. Elle fut initiée par le mathématicien britannique du milieu du XIX^{e} siècle **George Boole**.Aujourd'hui, l'algèbre de Boole trouve de nombreuses applications en informatique et dans la conception des circuits électroniques. Elle fut utilisée la première fois pour les circuits de commutation téléphoniques par **Claude Shannon***.

La théorie de BOOLE est une théorie de **classe** comme l'ensemble de tous les éléments partageant un même nom ou une même caractéristique. Ceci convenu, en imaginant deux classes A et B quelconques, Boole définit trois opérations fondamentales de l'esprit susceptible de s'exercer sur elles:

- Le **PRODUIT LOGIQUE**, que nous noterons A x B (ou A . B, ou encore AB), définissant **la classe des éléments obéissant simultanément aux définitions des classes A et B**.En absence de toutes parenthèses explicites, le produit logique est prioritaire sur la somme logique. Ainsi, (A.B)+C peut donc également s'écrire A.B+C.
- La **SOMME LOGIQUE**, que nous noterons A + B, définissant **la classe des éléments obéissant à l'une ou l'autre des définitions des classes A et B**.
- La **COMPLEMENTATION**, que nous noterions $\bar{A}$(ou$\bar{B}$), définissant **la classe des éléments n'obéissant pas à la définition de la classe A (ou B)**.

Attention ! Bien que leur nom et leur symbole nous y poussent, **ne confondons pas la somme et le produit logiques avec la somme et le produit arithmétiques, tels que nous les connaissons.** Les premiers s'exercent sur des classes, les seconds sur des nombres.

Mais ne nous emballons pas et appuyons-nous pour continuer sur l'exemple des deux classes suivantes:

- La classe A définie comme "les chasseurs",
- La classe B définie comme "les myopes".

Chaîne d'information

Partant de ces deux simples définitions, nous pouvons très facilement, par le jeu de notre seule pensée, définir quatre autres classes:

- La classe A.B définie comme "les chasseurs myopes",
- La classe A+B définie comme "les chasseurs et les myopes".
- La classe $\bar{A}$ définie comme "les non-chasseurs",
- La classe $\bar{B}$ définie comme "les non-myopes".

Il existe un moyen très visuel de traduire ces concepts un peu abstraits: il consiste à considérer les classes de Boole comme des ensembles. Dès lors, les opérations fondamentales de Boole prennent des noms plus familiers à nos souvenirs scolaires:

- La somme logique de deux classes se traduit par l'**union** (∪) entre les deux ensembles correspondants,
- Le produit logique par l'**intersection** (∩),
- La complémentation par... **le complément**.

Axiomes de l'algèbre de BOOLE :

Axiomes	
Commutativité	$A + B = B + A$ et $A . B = B . A$
Associativité	$(A + B) + C = A + (B + C)$ et $(A . B) . C = A . (B . C)$
Distributivité	$(A + B) . C = (A . C) + (B . C)$ $(A . B) + C = (A + C) .(B + C)$
Elément vide	$A + 0 = A$
Un tout	$A * 1 = A$
Tiers exclu	$A + \bar{A} = 1$(les chasseurs et les non chasseurs : tout le monde quoi)
Principe de contradiction	$A * \bar{A} = 0$ (l'ensemble des personnes qui sont ni **chasseurs** ni **pas chasseurs** : personne quoi !)

Les théorèmes de l'algèbre booléenne

Théorèmes	
Unicité du complément	$A + \bar{A} = 1$ et $A * \bar{A} = 0$
Idempotence	$A + A = A$ et $A * A = A$
Eléments absorbants	$A + 1 = 1$ et $A * 0 = 0$
Involution	$\overline{\overline{A}} = A$
Absorption	$A + A . B = A$ et $A .(A + B) = A$
Unicité des compléments des constantes	$\bar{0} = 1$ et $\bar{1} = 0$

Chaîne d'information

Théorème de DE MORGAN

Ajoutons à cette liste deux théorèmes importants, dits **THEOREMES de DE MORGAN**, fruits des travaux du mathématicien anglais Augustus de Morgan [1806-1871], qui annoncent que:

- le complément d'une somme logique est égal au produit logique des compléments:

$$\overline{A + B} = \bar{A} \cdot \bar{B}$$

- le complément d'un produit logique est égal à la somme logique des compléments:

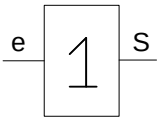
$$\overline{A \cdot B} = \bar{A} + \bar{B}$$

Rem : On démontre aussi que $A + \bar{A}B = A + B$ ou $\bar{A} + AB = \bar{A} + B$, plus généralement $A + \bar{A}(\ldots) = A + (\ldots)$.

4.1.2 – Les opérateurs logiques fondamentaux (oui, non, et, ou)

4.1.2.1 Fonction OUI :

Si on impose 1 en entrée, on observe 1 en sortie.
Si on impose 0 en entrée, on observe 0 en sortie.
L'équation de la sortie est donc : $S = e$

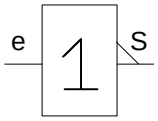


Rem : La fonction OUI n'a pas d'intérêt logiquement parlant, mais électriquement (comme les signaux logiques sont représentés par des niveaux physiques en électronique) deux intérêts sont à noter :

* La fonction OUI permet au signal électrique de se régénérer, fonction amplification.
* La fonction OUI introduit un retard dans la transmission du signal.

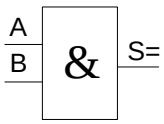
4.1.2.2 Fonction NON (aussi appelée INVERSEUSE)

Si on impose 1 en entrée, on observe 0 en sortie.
Si on impose 0 en entrée, on observe 1 en sortie.
Cette fonction se note « $\bar{}$ ».
L'équation de la sortie est donc : $S = \bar{e}$.
On lit « S égale e barre ».



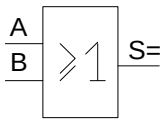
4.1.2.3 Fonction ET

Il faut imposer 1 à l'entrée **A ET** à l'entrée **B** pour observer **1** en sortie.
Cette fonction se note « ., *, ,».
L'équation de la sortie est donc : $S = A \cdot B = AB = A*B$,
*On lit « **S égale A ET B** ».*



4.1.2.4 Fonction OU

Il faut imposer **1** à l'entrée **A OU** à l'entrée **B** pour observer **1** en sortie.
Cette fonction se note « + ».
L'équation de la sortie est donc : ***S = A + B,***
on lit « **S égale A ou B** ».

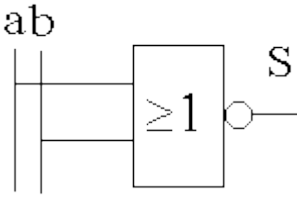


4.1.3 – Les autres opérateurs

4.1.3.1 Fonction NOR (Not OR, Ou Non, NI)

Equation : **S** = $\overline{a + b}$

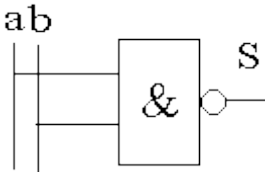
La sortie : **S** reproduit le complément de la somme logique des variables d'entrée **a** et **b**. La fonction **NOR** (ou NI) est une fonction universelle, c'est à dire que les fonctions OUI, NON, OU, ET peuvent être réalisées avec uniquement des opérateurs NOR.



4.1.3.2 Fonction NAND (Not AND, Et Non)

Equation : **S** = $\overline{a \cdot b}$

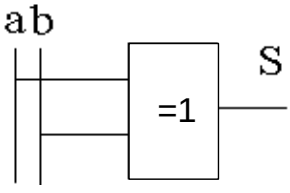
La sortie : **S** reproduit le complément du produit logique des variables d'entrée **a** et **b**. La fonction **NAND** est aussi une fonction universelle.



4.1.3.3 Fonction XOR (ou exclusif)

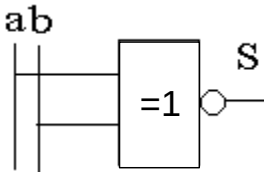
Equation: $S = a\bar{b} + \bar{a}b = a \oplus b$

La sortie : **S** sera active (niveau 1) si l'une ou l'autre des variables **a** ou **b** est active, la sortie sera dans le niveau bas (0) si les deux variables sont dans le même état.



4.1.3.4 Fonction XNOR (ou exclusif complémentée)

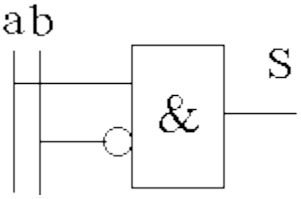
Equation : $S = \bar{a} \bullet \bar{b} + a \bullet b = \overline{\alpha \oplus b}$



La sortie : **S** sera active (niveau 1) si les deux variables sont dans le même état, la sortie sera dans l'état bas si les variables sont dans des états différents.

Rem : on trouve aussi une fonction nommée **Inhibition**

Equation : **S = a . $\bar{b}$**



La sortie **S** reproduit le produit logique d'une variable d'entrée a et du complément d'une variable d'entrée b, la variable b autorise (état bas) ou non (état haut) la sortie **S** à prendre la valeur de la variable **a** autrement dit **S = a si b = 0**.

4.1.4 – Les Différents modes de représentation

Il existe plusieurs façon de représenter une fonction logique, dans le tableau suivant on présente respectivement les représentations :

Symbolique (Européen), c'est le dessin du symbole qui représente la fonction logique, il existe aussi une représentation américaine (voir tableau schéma à contacts p : 46).

Equation : L'équation de la sortie sous forme algébrique, S= f(e_i).

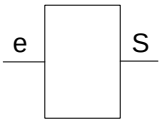
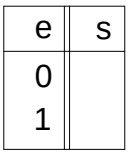
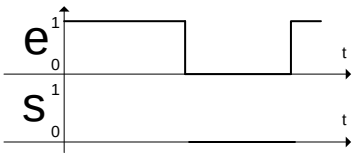
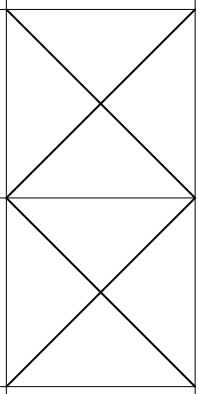
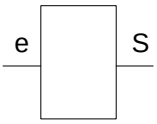
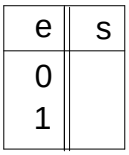
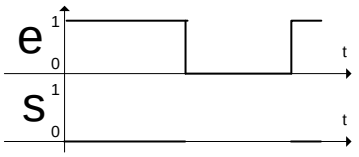
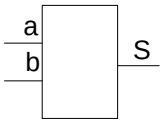
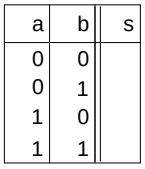
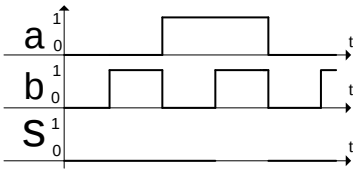
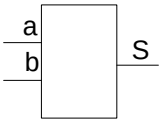
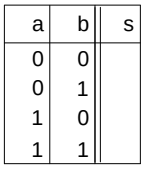
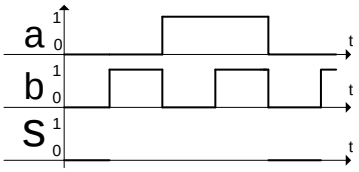
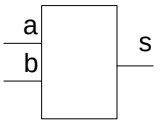
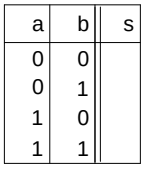
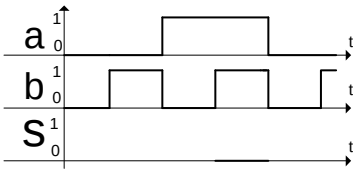
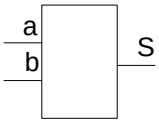
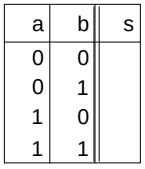
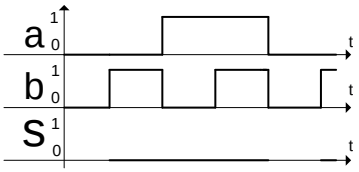
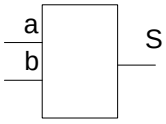
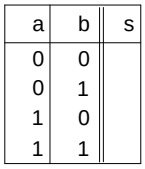
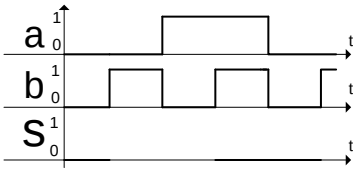
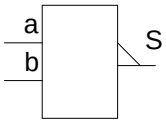
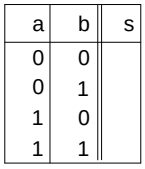
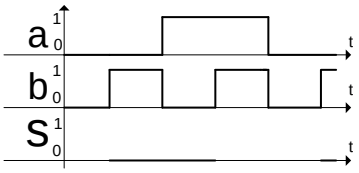
Table de vérité : Tableau qui met en relation les différentes combinaisons du vecteur d'entrée et l'état de(s) la sortie(s) de cette fonction logique.

Chronogramme : Représentation temporelle de l'évolution de la sortie en fonction de l'état des différentes variables d'entrées.

Priorité : La priorité traduit un état prioritaire en entrée qui force la sortie à prendre un état déterminé, ceci indépendamment des autres variables d'entrées.

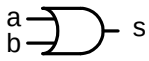
Schéma à contacts : c'est un schéma électrique constitué d'interrupteurs (NO normalement ouvert, NF normalement fermé « pour une variable complémentée ») et d'une lampe symbolisant la sortie, si la lampe est allumée la sortie est à « 1 », si elle est éteinte la sortie est à « 0 », raisonnement en logique positive.

Chaîne d'information

Fonction de l'opérateur	Symbole Européen	Equation	Table de vérité	Chronogramme	Priorité
OUI		S =			
NON		S =			
ET		S=			
OU		S =			
NAND		S=			
NOR		S=			
OU EXCLUSIF		S=			
OU EXCLUSIF COMPLEMENTEE		S=			

Chaîne d'information

Rem : L'alimentation des schémas à contacts est notée V+ et V- : Exemple **V+ = 5v** et **V- = 0v** (gnd : ground)

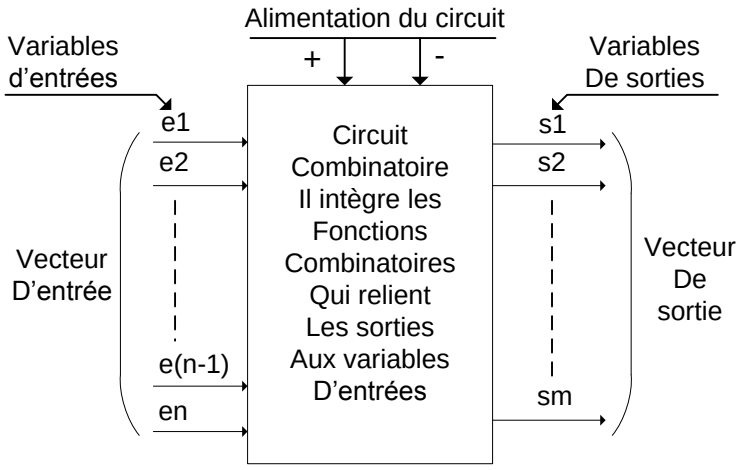
Fonction de l'opérateur	Symbole Américain	Equation	Schéma à contacts
OUI		$S = e$	
NON		$S = \bar{e}$	
ET		$S = a.b$	
OU		$S = a+b$	
NAND		$S = \bar{a}+\bar{b}$	
NOR		$S = \bar{a}.\bar{b}$	
OU EXCLUSIF XOR		$S=\bar{a}b+a\bar{b}$	
OU EXCLUSIF COMPLEMENTEE XNOR		$S=\bar{a}\bar{b}+ab$	

4.2 – La logique combinatoire

4.2.1 – Présentation

L'étude de la logique combinatoire permet de concevoir des circuits combinatoires.

Représentation d'un circuit combinatoire :



Ce circuit combinatoire est représentatif de tous les types de circuit, du plus simple au plus complexe. Le vecteur d'entrées possède de 1 à n variables d'entrées (logique), le vecteur de sortie possède de 1 à m variables de sorties (logique). On peut remarquer qu'il n'existe pas de relation (connexion) d'une (ou plusieurs) sortie(s) vers le vecteur d'entrée, on dit que c'est un circuit en boucle ouverte. Autrement dit à une combinaison des variables d'entrée correspond un vecteur (et un seul) de sortie.

On peut classer les circuits combinatoires en de catégories, les circuits câblés et les circuits programmables. Il existe un grand nombre de circuits combinatoires câblés (sous forme de circuits intégrés, CI) réalisant les fonctions combinatoires de bases, on peut citer :

- les opérateurs logiques fondamentaux (oui, non, and, or, nand, nor, xor, xnor)
- Les Multiplexeurs
- Les démultiplexeurs
- Les comparateurs
- Les encodeurs, les décodeurs, les transcodeurs
- Les fonctions arithmétiques de base (addition, soustraction, multiplication, division)
- Les générateurs de parité
- Etc…

Malgré la diversité de circuits disponibles dans le commerce, pour une application spécifique il est parfois nécessaire de concevoir son propre circuit combinatoire pour répondre à un besoin précis. Il est donc nécessaire d'établir les équations logiques qui régissent le fonctionnement du circuit combinatoire.

Chaîne d'information

L'étude et la recherche des équations de fonctionnement du circuit nécessitent une méthode rigoureuse qui permette de réaliser un circuit optimisé (équations simplifiées) qui garantissent la bonne solution au problème à résoudre, tant d'un point de vue statique que dynamique (aléa de fonctionnement, phénomène transitoire). L'étude commence par l'établissement d'un cahier des charges qui explique littéralement le fonctionnement du circuit (que l'on veut réaliser), on traduit ensuite ce cahier des charges en une table de vérité qui met en relation les entrées et les sorties du circuit.

La table de vérité traduit l'état du vecteur de sortie en fonction d'une combinaison des variables d'entrées. On traduit ensuite la table de vérité en équations logiques : pour chaque sortie on exprime les différentes combinaisons du vecteur d'entrée qui active cette sortie (on parle de raisonnement en logique positive si on considère une sortie active au niveau 1) que l'on simplifiera.

La simplification d'une équation logique met en œuvre les théorèmes de l'algèbre de BOOLE. Il existe une méthode graphique qui permet de simplifier une équation logique si le nombre de variable d'entrée n'excède pas quatre variables : La méthode des tableaux de **KARNAUGH**. Dans les autres cas (+ de 4 variables d'entrées), il faudra simplifier les équations sous forme algébrique « Méthode algébrique toujours possible mais démarche intuitive qui dépend de l'habileté et de l'expérience », Le logiciel « **LOGICALC** » permet de simplifier des équations logiques jusqu'à 6 variables.

Lorsque l'on a obtenu les équations simplifiées de la fonction combinatoire deux solutions sont possibles pour sa réalisation :

- On peut câbler cette fonction en utilisant des circuits intégrés vendus dans le commerce.
- On peut programmer cette fonction dans un circuit programmable.

Question : quel est l'intérêt de réaliser un circuit combinatoire (câblé ou programmé) ? Un calculateur peut se charger par programme de réaliser la même fonction.

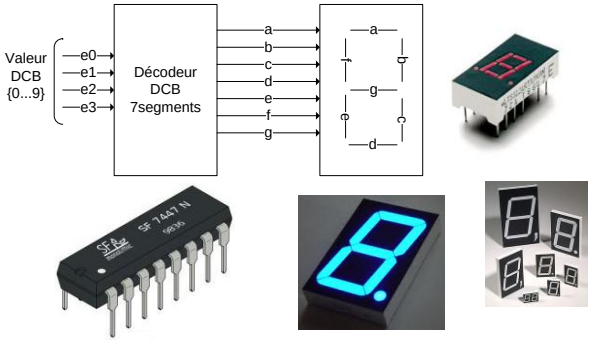
Réponse :

- Décharger le calculateur de cette tâche, surtout s'il a beaucoup d'autres choses à faire.
- Un temps d'exécution beaucoup plus rapide (par rapport à l'exécution d'un programme), on dit qu'un circuit câblé ou programmé travail à la vitesse de la matière. Cela est quelque fois nécessaire pour des applications nécessitant un temps de réponse rapide.

4.2.2 – Etude ducircuit décodeur DCB/7segments

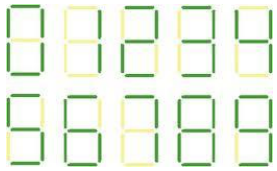
Nous allons maintenant faire l'étude et le câblage d'un circuit combinatoire afin d'identifier toutes les étapes de sa réalisation.

1 : Présentation: Un décodeur DCB/7segments permet l'affichage d'un chiffre décimal sur un afficheur 7segments.



Chaîne d'information

2 : Cahier des charges: Ici le cahier des charges et simple, le circuit réalise la conversion d'un mot binaire 4bits codé en DCB en la commande des différents segments nécessaires à l'obtention de l'affichage du chiffre désiré suivant la règle suivante :



3 : Table de vérité:

	D	e3	e2	e1	e0	a	b	c	d	e	f	g
	0	0	0	0	0							
	1	0	0	0	1							
	2	0	0	1	0							
	3	0	0	1	1							
	4	0	1	0	0							
	5	0	1	0	1							
	6	0	1	1	0							
	7	0	1	1	1							
	8	1	0	0	0							
	9	1	0	0	1							
Impossible En DCB	X	1	0	1	0							
	X	1	0	1	1							
	X	1	1	0	0							
	X	1	1	0	1							
	X	1	1	1	0							
	X	1	1	1	1							

X= 0 ou 1, on peut choisir puisque normalement
Ces combinaisons n'existent pas

4 : Recherche des équations logiques des segments:

Pour obtenir l'équation logique de la commande d'un segment, il faut prendre en compte toutes les combinaisons du vecteur d'entrée qui activent le segment. On considère ici le segment actif s'il est allumé c'est un raisonnement en logique positive.

Rem : Convention d'écriture une variable d'entrée e_i **complémentée se notera : /**e_i

Cas du Segment e:

Les différentes combinaisons du vecteur d'entrée (il en existe 4) qui active cette sortie sont:

Le segment **e = 1** si **e0=e1=e2=e3=0** qu'on traduit en : **e= /e0. /e1. /e2. /e3 = 1 (1.1.1.1=1)**
OU Le segment **e = 1** si **e0=e2=e3=0, e1=1** qu'on traduit en : **e= /e0. e1. /e2. /e3 = 1**
OU Le segment **e = 1** si **e0=e3=0, e1=e2=1** qu'on traduit en : **e= /e0. e1. e2. /e3 = 1**
OU Le segment **e = 1** si **e0=e1=e2=0, e3=1** qu'on traduit en : **e= /e0. /e1. /e2. e3 = 1**
Les quatre combinaisons qui activent la sortie s'appelle des **Mintermes** (lire : mine terme).

On obtient l'équation générale du segment e :

e = /e0. /e1. /e2. /e3 + /e0. e1. /e2. /e3 + /e0. e1. e2. /e3 + /e0. /e1. /e2. e3

Chaîne d'information

Rem : Si on programme un circuit pour réaliser ce décodeur DCB/7segments on pourra laisser l'équation sous cette forme, par contre si on câble l'équation avec des opérateurs logiques fondamentaux (non, et, ou, nand, nor), il faut simplifier cette équation.

Simplification de l'équation avec les théorèmes de l'algèbre de BOOLE :

« *Méthode algébrique toujours possible mais démarche intuitive qui dépend de l'habileté et de l'expérience*»

Démonstration :

$$e = \overline{e0}.\overline{e1}.\overline{e2}.\overline{e3} + \overline{e0}.e1.\overline{e2}.\overline{e3} + \overline{e0}.e1.e2.\overline{e3} + \overline{e0}.\overline{e1}.\overline{e2}.e3$$

$$e = \overline{e0}.\overline{e2}.\overline{e3}\underbrace{(\overline{e1} + e1)}_{=1} + \overline{e0}.e1.e2.\overline{e3} + \overline{e0}.\overline{e1}.\overline{e2}.e3$$

$$e = \underbrace{\overline{e0}.\overline{e2}.\overline{e3} + \overline{e0}.e1.e2.\overline{e3}} + \overline{e0}.\overline{e1}.\overline{e2}.e3$$

$$e = \overline{e0}.\overline{e3}\underbrace{(\overline{e2} + e2.e1)} + \overline{e0}.\overline{e1}.\overline{e2}.e3$$

$$\overline{e2} + e1$$

$$e = \overline{e0}.\overline{e3}(\overline{e2} + e1) + \overline{e0}.\overline{e1}.\overline{e2}.e3$$

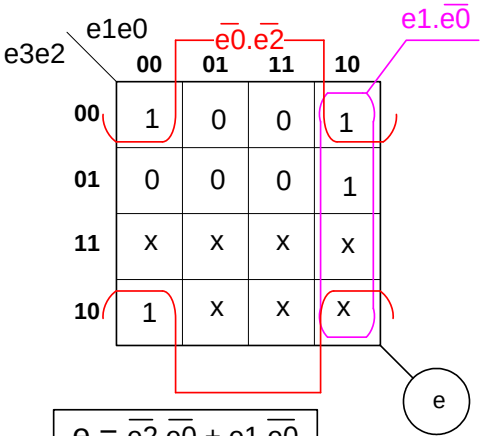
$$e = \underbrace{\overline{e0}.\overline{e2}.\overline{e3}} + \overline{e0}.e1.\overline{e3} + \underbrace{\overline{e0}.\overline{e1}.\overline{e2}.e3}$$

$$e = \overline{e0}.\overline{e2}\underbrace{(\overline{e3} + e3.\overline{e1})} + \overline{e0}.e1.\overline{e3}$$

$$\overline{e3} + \overline{e1}$$

$$\boxed{e = \overline{e0}.\overline{e2}.\overline{e3} + \overline{e0}.\overline{e1}.\overline{e2} + \overline{e0}.e1.\overline{e3}}$$

***Simplification de l'équation par la méthode des tableaux de* KARNAUGH :**



$$\boxed{e = \overline{e2}.\overline{e0} + e1.\overline{e0}}$$

$$\boxed{e = \overline{e0}(\overline{e2} + e1)}$$

On peut remarquer que la solution obtenue par cette méthode est plus simple, la méthode des tableaux de KARHAUGH permet en effet d'obtenir la forme la plus concise de l'équation de la sortie (moins évident lorsqu'on simplifie avec les théorèmes !), c'est une méthode graphique en deux dimensions qui présente les différentes combinaisons du vecteur d'entrée en code GRAY (ou binaire réfléchi) nous l'étudierons plus tard dans l'étude des capteurs numériques.

Chaîne d'information

Cas du Segment c:

Sur la table de vérité du décodeur DCB/7segments la sortie commandant le segment « **c** » est active pour toute les combinaisons du vecteur d'entrée sauf pour le chiffre « 2 ». Pour obtenir l'équation du segment « **c** », il est plus simple de rechercher l'équation de la **« non activité »** de ce segment, quand **c=0** (une seule combinaison du vecteur d'entrée) et dans prendre le complément qui nous donnera donc l'équation pour **c=1**.
La combinaison du vecteur d'entrée qui n'active pas le segment « c » est:

Le segment **c=0** si **e0=e2=e3=0, e1=1** qu'on traduit en : **/c= /e0. e1. /e2. /e3**

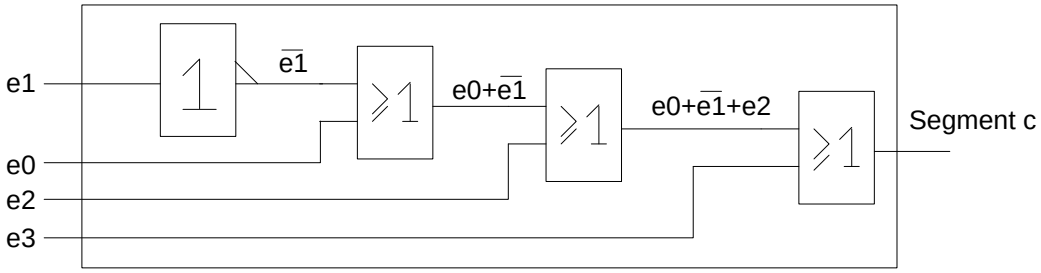
On obtient l'équation générale du segment c :

$$\overline{c} = \overline{e0}.e1.\overline{e2}.\overline{e3}$$

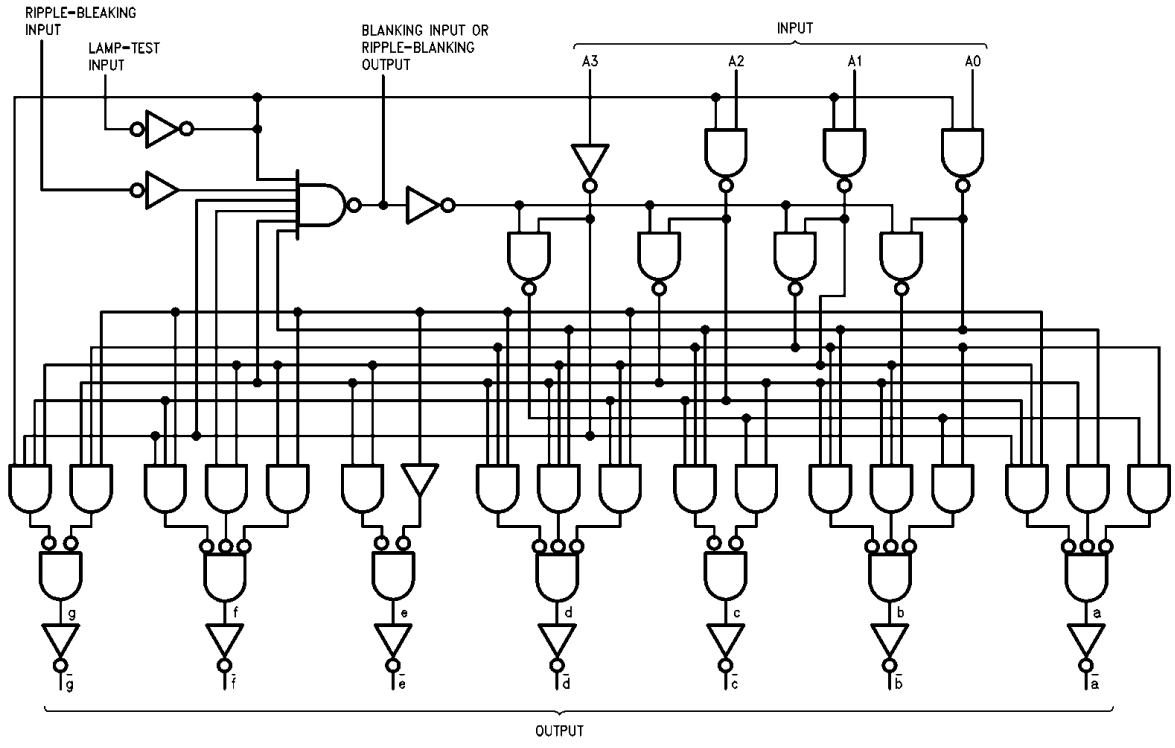
$$c = \overline{\overline{c}} = \overline{\overline{e0}.e1.\overline{e2}.\overline{e3}}$$ Théorème de DE MORGAN

$$c = e0+\overline{e1}+e2+e3$$

5 : Câblage dusegment« c »



Voici le schéma interne du décodeur DCB/7segments vendu dans le commerce Réf : 74LS47



Chaîne d'information

4.2.3 – Exercices

4.2.3.1 – Applications du théorème de DE MORGAN :

$S_1 = \overline{(a + \bar{a}b)(bc + \bar{b})}$

$S_2 = \overline{(a + ab)(b + \bar{b}a)}$

*** Simplifier les équations suivantes :**

$S_3 = ab + \bar{a}b + a\bar{b}c + \bar{c}\bar{b}$

$S_4 = a(\bar{c}+ab\bar{c}) + (a\bar{b}c+ac)$

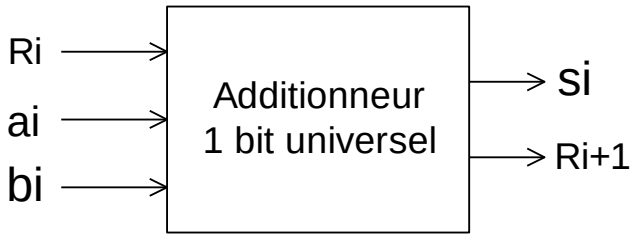
4.2.3.2 – Modes de représentations :

*** Remplir le tableau suivant :**

Equation	Table de verité	Schéma à contacts
S = a + b + c	a b c s 0 0 0 0 0 1 0 1 0 0 1 1 1 0 0 1 0 1 1 1 0 1 1 1	V+ S V-
S = a b c	a b c s 0 0 0 0 0 1 0 1 0 0 1 1 1 0 0 1 0 1 1 1 0 1 1 1	V+ S V-
S = ab+bc+ac	a b c s 0 0 0 0 0 1 0 1 0 0 1 1 1 0 0 1 0 1 1 1 0 1 1 1	V+ S V-

4.2.3.3 – Etude et réalisation d'un additionneur 1bit universel :

Problème : On désire réaliser un ***additionneur 1 bit universel.***



Présentation :

L'additionneur 1 bit universel permet d'additionner deux variables binaires a et b. Il est universel par le fait qu'il peut être utilisé sur tous les rangs d'un nombre binaire. Ainsi nous étudierons dans un premier temps comment le réaliser puis nous nous serviront de nos résultats pour créer un additionneur 8 bits, addition de deux mots binaires A et B.

Définitions :

ai : variable binaire définie au rang i du mot A
bi : variable binaire définie au rang i du mot B
si : le résultat au rang i de la somme **ai+bi=si**
Ri : retenue provenant du rang inférieur (i-1)
Ri+1 : retenue générée par l'additionneur de rang i pour le rang supérieur (i+1).

*Analyse :*Si on considère l'addition de 2 variables binaires (ai et bi), le résultat obtenu est :

Si ai=0 et bi=0 la somme est nulle donc si=0 et Ri+1=0
Si ai=0 et bi=1 la somme est '1' donc si=1 et Ri+1=0
Si ai=1 et bi=0 la somme est '1' donc si=1 et Ri+1=0
Si ai=1 et bi=1 la somme est '1' donc si=0 et Ri+1=1

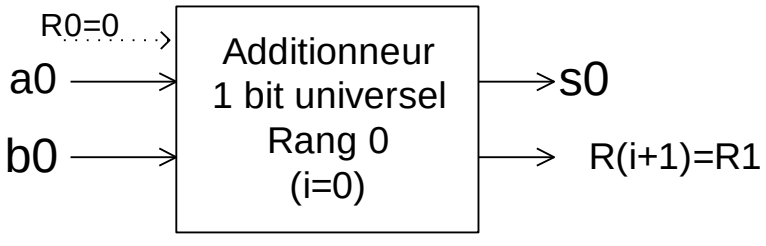
On constate que l'addition des deux variables binaires génère une retenue si ai=bi=1 alors Ri+1=1.

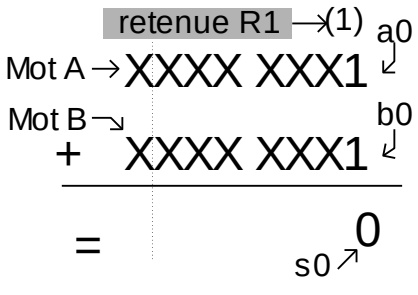
Hypothèse : considérons le rang zéro *: (i = 0)*

La retenue R0 au rang zéro n'est pas prise en compte si on additionne des nombres entiers, il n'y a pas de rang -1 qui aurait pu générer une retenue au rang zéro, il faudra donc la fixer à zéro pour quelle n'influence pas les calculs des autres additionneurs 1 bit (R0=0).

L'addition des variables a0 et b0 donne s0 et une retenue éventuelle sur R1.

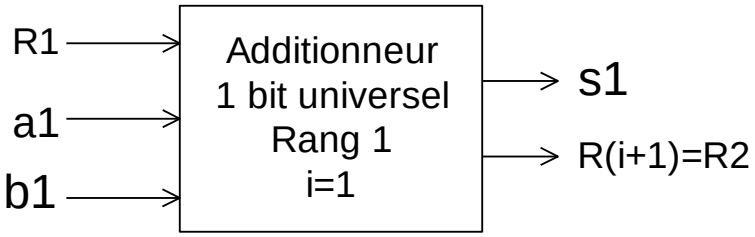
Cette retenue sera à transmettre à l'additionneur de rang 1.





Chaîne d'information

Hypothèse : considérons le rang 1 *: (i = 1)*
L'additionneur de rang 1 doit prendre en compte une variable binaire d'entrée supplémentaire : *La retenue éventuellement* générée par le rang zéro.
On obtient donc la représentation suivante (conforme à la représentation de l'additionneur universel 1 bit soit 3 entrées et 2 sorties) :



Pour le rang 1 et pour les rangs suivants l'additionneur aura toujours la même structure : 3 entrées et 2 sorties.

Soit à réaliser l'additionneur 1 bit universel définit par la table d' entrées-sorties suivante :

1 : Compléter le tableau suivant en respectant la fonction 'additionneur'

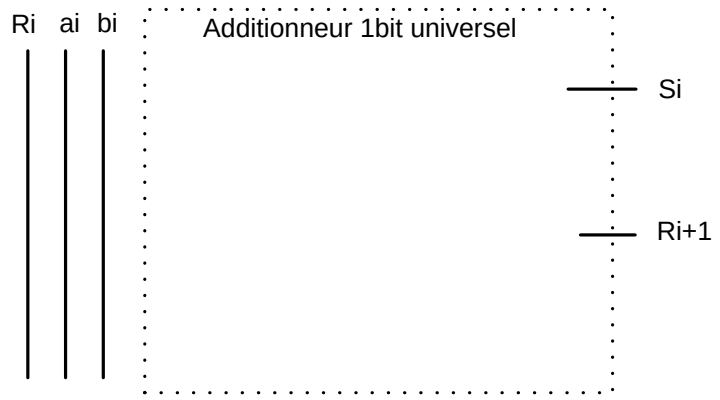
Ri	ai	bi	si	R_{i+1}
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

2 : Recherche des équations des sorties Si et R_{i+1}

S_i =
R_{i+1} =

3 : Simplifier les équations des sorties Si et R_{i+1}

4 : Câbler les équations des sorties Si et R_{i+1}

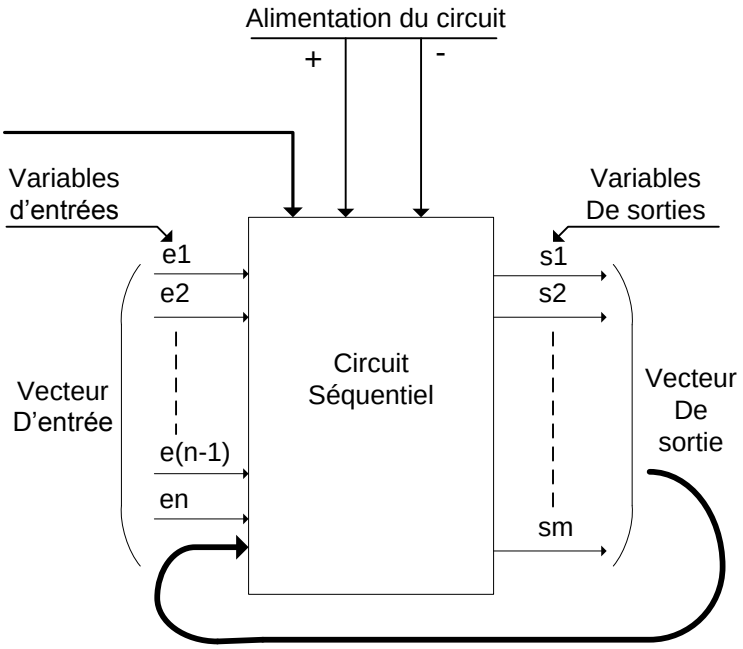


Rem : en réalisant 8 fois la même structure on réalise un additionneur 8bits

4.3 – La logique séquentielle

4.3.1 – Présentation

Représentation d'un circuit séquentiel:



On peut remarquer par rapport au circuit combinatoire qu'un circuit séquentiel possède une rétroaction des sorties vers les entrées (on parle de boucle fermée) et qu'il a besoin d'une variable d'entrée fondamentale **LE TEMPS**.

Chaîne d'information

L'étude des circuits séquentiels permet d'aborder le fonctionnement des machines à états. Dans la description SYSML c'est le diagramme d'états transitions qui permettra de schématiser la description du fonctionnement d'un système.

Les circuits séquentiels fondamentaux sont les bascules « briques élémentaires ». L'association de bascules et d'opérateurs logiques (ch : logique combinatoire) permettra de réaliser des circuits séquentiels plus complexes jusqu'à obtenir la structure d'un calculateur (processeur, microcontrôleur, DSP,…).

4.3.2 – Etude des bascules

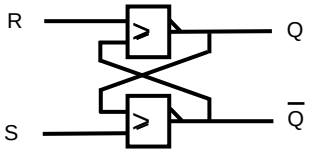
1: Introduction

La bascule est une mémoire élémentaire, elle permet de stocker l'état d'une variable logique. Il existe plusieurs types de bascules (RS, /R/S, D, JK,…). **La bascule** est un **circuit bistable** pouvant prendre deux états logiques sur sa sortie: **0 et 1**.L'état de la bascule peut être modifié en agissant sur une ou plusieurs entrées.

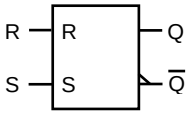
Le nouvel état de la bascule **dépend de l'évolution des entrées et de l'état précédent des sorties**. **C'est le principe fondamental utilisé pour les circuits séquentiels.**

2 : BASCULE RS à opérateur NON-OU (NOR)

2.1 SCHEMA DE PRINCIPE

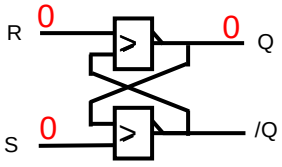
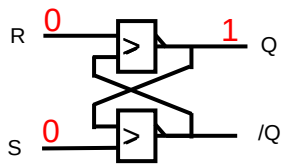
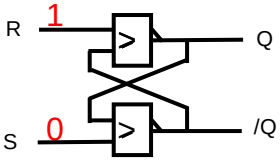
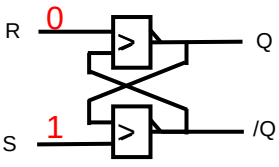
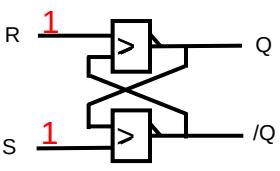


2.2 SYMBOLE



2.3 EXPLICATION

A partir de la table de vérité de la fonction NON OU vérifier pour les différents cas, l'état de la sortie des portes logiques qui composent la bascule

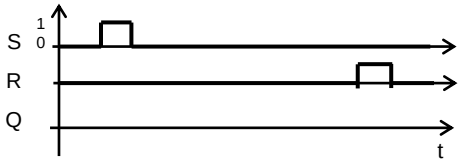
R=0 S=0	R=1 S=0	R=0 S=1	R=1 S=1
1er cas = Q était à 0  2ème cas = Q était à 1 			

Chaîne d'information

2.4 TABLE DE VERITE

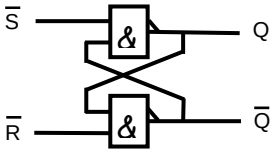
S	R	Q	/Q	
0	0	Q_n	Q_n	Aucun changement
0	1	0	1	
1	0	1	0	
1	1	1	1	Etat interdit

2.5 CHRONOGRAMME

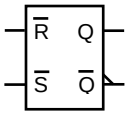


3. BASCULE /R/S à opérateurs NON-ET (NAND)

3.1 SCHEMA DE PRINCIPE

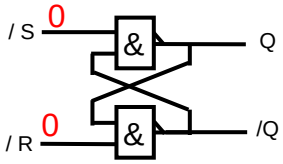
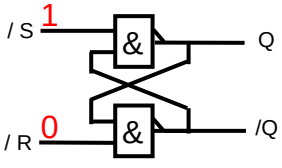
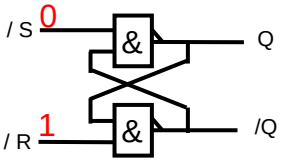
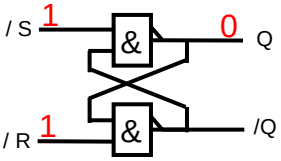
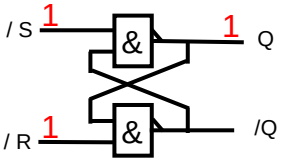


3.2 SYMBOLE



3.3 EXPLICATION

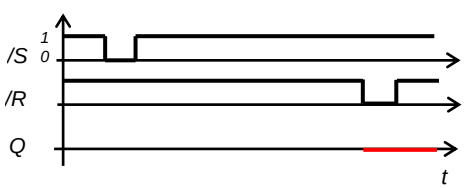
A partir de la table de vérité de la fonction NON ET, vérifier pour les différents cas l'état de la sortie des portes logiques qui composent la bascule.

/R=0 /S=0	/R=0 /S=1	/R=1 /S=0	/R=1 /S=1
			1er cas = Q était à 0  2ème cas = Q était à 1 

3.4 TABLE DE VERITE

/R	/S	Q	/Q	
0	0	0	0	Etat interdit
0	1	0	1	
1	0	1	0	
1	1	Q_n	Q_n	Aucun changement

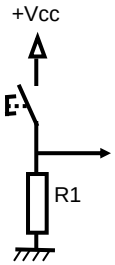
3.5 CHRONOGRAMME

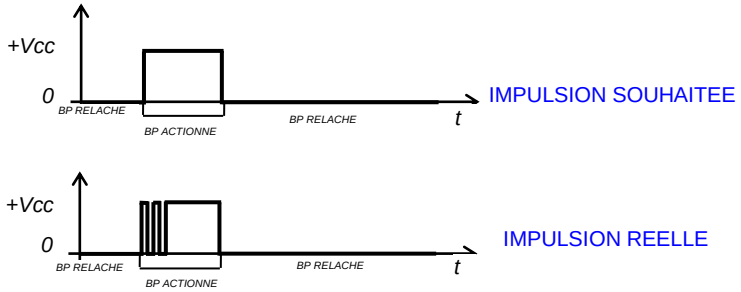


Chaîne d'information

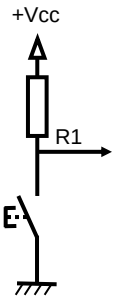
APPLICATION : Mise en œuvre d'une BASCULE RS dans un circuit anti rebond d'un contact

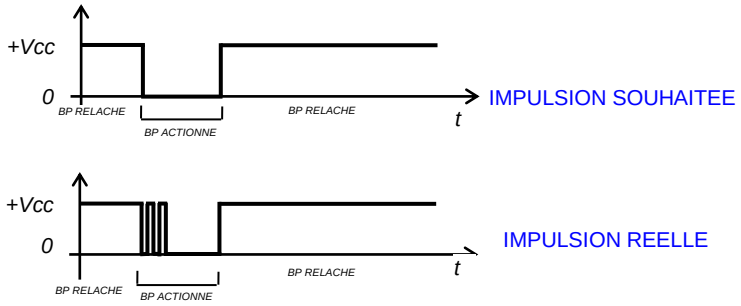
Envoyer un niveau haut (Front montant) :





Envoyer un niveau bas (Front descendant) :



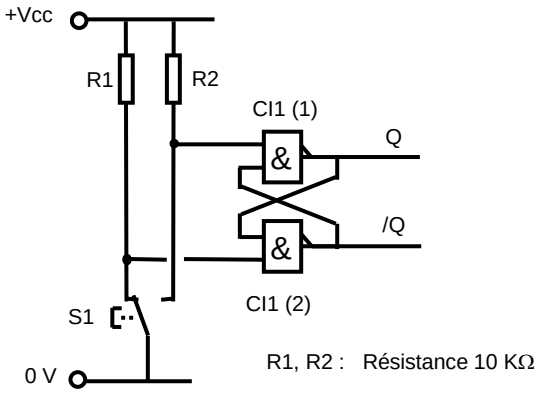


A chaque manœuvre du bouton poussoir BP, il y a rebondissement du contact en cuivre, avant que le contact ne soit maintenu définitivement. Pour un contact ordinaire, il peut y avoir jusqu'à une dizaine de rebonds pendant quelques millisecondes. Pour la commande d'organes ayant une constante de temps assez élevée, ces rebonds ne nuisent pas au fonctionnement (relais, lampes, moteurs etc…), par contre dans le cas de circuits rapides (ex: circuits intégrés en électronique qui réagissent en quelques nanosecondes), il est nécessaire de filtrer les rebonds, afin de ne pas prendre en compte les états intempestifs. Tout se passe donc, avec les rebonds, comme si l'on agissait plusieurs fois sur le BP au lieu d'une.

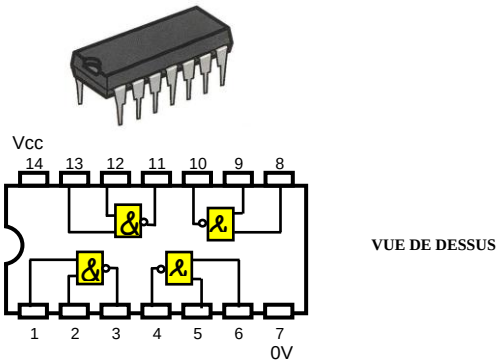
Il est donc nécessaire d'utiliser un dispositif permettant de supprimer ces rebonds. La bascule mémorise les informations dès leurs premières apparitions.

Chaîne d'information

SCHEMA DE PRINCIPE



BROCHAGE DU CIRCUIT INTEGRE 4011



VUE DE DESSUS

Ce circuit permet de ne transmettre qu'une seule impulsion en sortie Q. Quand l'inverseur rebondi, il se situe entre les 2 pôles. R et S sont reliés au +Vcc (R=S=1), donc la bascule ne change pas d'état. C'est l'état mémoire de la bascule, les rebonds des contacts ne sont pas pris en compte.

BASCULES EN MODE ASYNCHRONE OU SYNCHRONE

Il existe deux types de fonctions séquentielles :

- Les fonctions séquentielles **asynchrones** dont les sorties peuvent évoluer dès que les entrées changent (nous venons de les voir précédemment).
- Les fonctions séquentielles **synchrones** dont les sorties évoluent selon les entrées mais seulement après un ordre de synchronisation appelé signal d'horloge (nous allons les découvrir ci-après).

Dans une **bascule asynchrone**, les ordres appliqués sur les entrées sont pris en compte immédiatement pouvant provoquer immédiatement en sortie le changement d'état correspondant.

Dans une **bascule synchrone**, l'exécution de l'ordre n'intervient qu'avec un signal de synchronisation appelé **horloge (H)** ou **Clock (C)**.

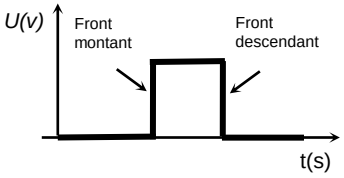
Travail sur niveau ou sur front

Un **circuit travaille sur un niveau** lorsque les informations sont prises en compte après la **stabilisation haute ou basse du signal** de validation ou d'horloge.

Un **circuit travaille sur front** lorsque la prise en compte des informations se fait au moment de la **montée ou de la descente du signal** de validation.

Dans les tables de vérité, le front est identifié par les symboles :

FRONT MONTANT **FRONT DESCENDANT**



Chaîne d'information

Les circuits qui suivent travailleront, sur des niveaux, des fronts montants, ou des fronts descendants. Un circuit qui travaille sur un front aura le symbole suivant sur l'entrée d'horloge :

Horloge agissant pendant la durée d'un niveau haut (1) H

Horloge agissant sur un front montant H

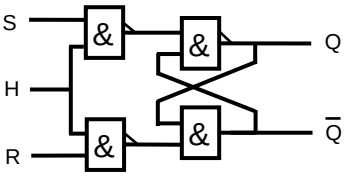
Horloge agissant pendant la durée d'un niveau bas (0) H

Horloge agissant sur un front descendant H

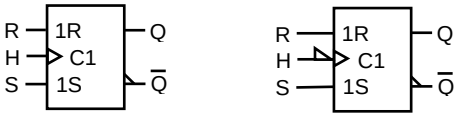
4. BASCULE RSH (Bascule synchrone)

C'est une bascule RS mais dont on a synchronisé les sorties avec des **impulsions d'horloge. Le changement d'état peut** s'effectuer sur niveau 0 ou 1 et sur front montant ou descendant.

4.1 SCHEMA DE PRINCIPE



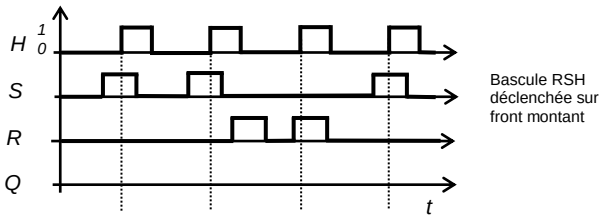
4.2 SYMBOLE



4.3 TABLE DE VERITE

H	S	R	Q	
0	X	X		Etat précédent
↑ ou ↓	0	0		Etat précédent
↑ ou ↓	0	1		
↑ ou ↓	1	0		
↑ ou ↓	1	1		

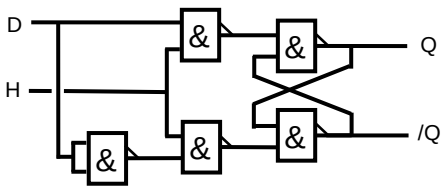
4.4 CHRONOGRAMME



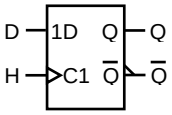
5. BASCULE D

Ces mémoires ne possèdent qu'une seule entrée D. C'est une bascule dont les entrées de prédétermination sont contrôlées par un inverseur ce qui évite toute possibilité d'indétermination.

5.1 SCHEMA DE PRINCIPE



5.2 SYMBOLE



A partir d'une bascule RSH on commande les entrées par le même signal en utilisant un inverseur.

Chaîne d'information

5.3 TABLE DE VERITE

D	H	Q
X	0	
X	1	
0	⎍	
1	⎍	

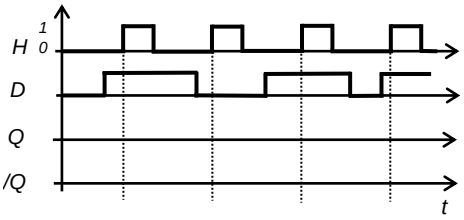
Etat précédent

Etat précédent

⎍ Front montant

X : Etat logique indifférent 0 ou 1

5.4 CHRONOGRAMME



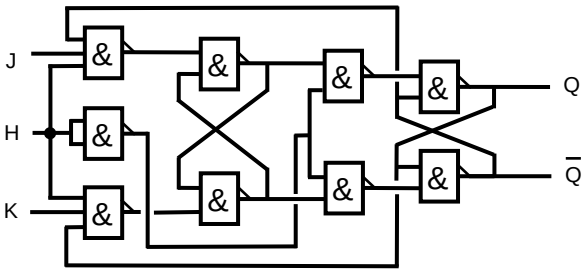
Ce type de bascule change d'état uniquement sur le **front montant** du signal d'horloge. On transmet ainsi la valeur de D en Q.

6. BASCULE JK

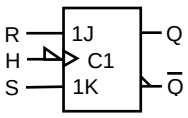
On l'appelle aussi « bascule maître-esclave », car lors de sa création, son fonctionnement rappelait les règles d'un jeu de cartes anglo-saxon. De là l'origine des lettres désignant ses entrées J et K et la sortie Q.

J = JACK (Le valet)
K = KING (le roi)
Q = QUEEN (la dame)

6.1 SCHEMA DE PRINCIPE



6.2 SYMBOLE



6.3 TABLE DE VERITE

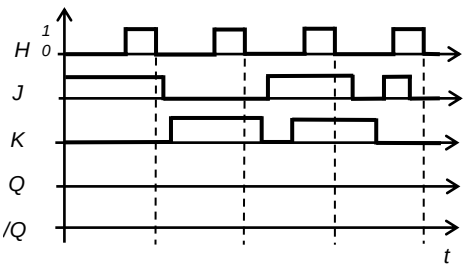
H	J	K	Q
X	X	X	
⎍	0	0	
⎍	0	1	
⎍	1	0	
⎍	1	1	

Aucun changement

Basculement à chaque front descendant du signal d'horloge

X : Etat logique indifférent 0 ou 1

6.4 CHRONOGRAMME

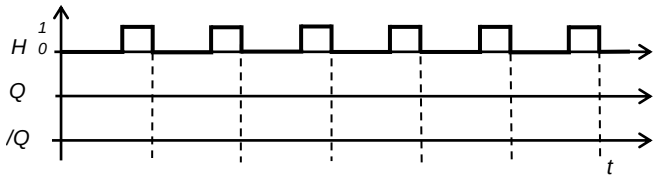


C'est maintenant les fronts descendants du signal d'horloge qui sont actifs.
Si J et K sont différents, J impose son niveau à la sortie Q
Si J et K sont à zéro la sortie reste inerte.
Si J et K sont à 1, Q change de niveau à chaque front descendant du signal d'horloge.

Chaîne d'information

6.5 CHRONOGRAMME POUR J = K = 1

La sortie Q change d'état à chaque impulsion d'horloge.



Il est très facile de maintenir J et K à 1 en permanence en le reliant au +Vcc.
On s'aperçoit sur le chronogramme que dans ce cas, **la bascule JK** **les impulsions d'horloge**

La **bascule** est l'élément de base de la logique séquentielle. En effet, en assemblant des bascules, on peut réaliser des **compteurs**, des **registres**, des **registres à décalage**, des **mémoires**.

4.3.3– Exercice

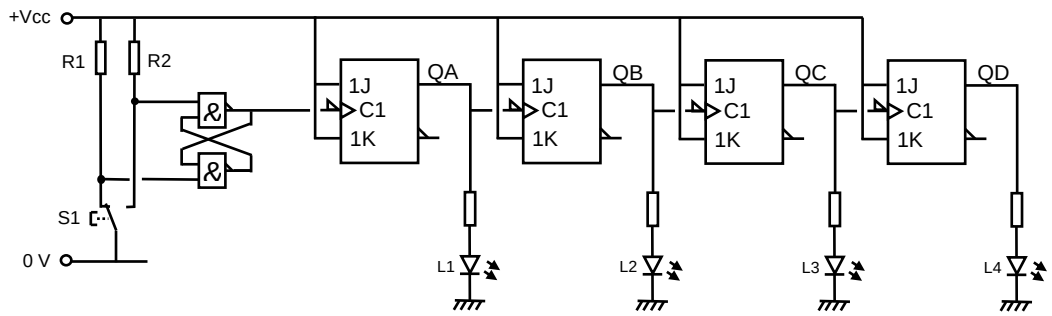
APPLICATION : COMPTEUR AVEC BASCULES JK

Principe

Si l'on place deux bascules JK montées en cascade, chaque bascule effectuant une division par deux, on aura une division par quatre (2^2 = 4), avec trois bascules en cascade on aura une division par huit (2^3 = 8), avec quatre bascules en cascade on aura une division par seize (2^4 = 16). Avec 4 bascules on obtient 16 combinaisons binaires différentes soit en binaire pur de 0000 à 1111 (de 0 à 15 en décimal).

Schéma de principe

On utilise quatre bascules JK dont les entrées J et K sont reliées au +Vcc (J=K=1). Le signal d'horloge CLOCK est envoyé grâce à un bouton poussoir muni d'un circuit anti rebond. La visualisation des sorties de chaque bascule est réalisée par 4 diodes électroluminescentes.



Chaîne d'information

Chronogramme

En observant ce chronogramme on remarque que la sortie de chaque bascule délivre une valeur binaire. Le mot binaire correspondant à l'ensemble des sorties des bascules (QA, QB, QC, QD) correspond au nombre (décimal) d'impulsions d'horloge.

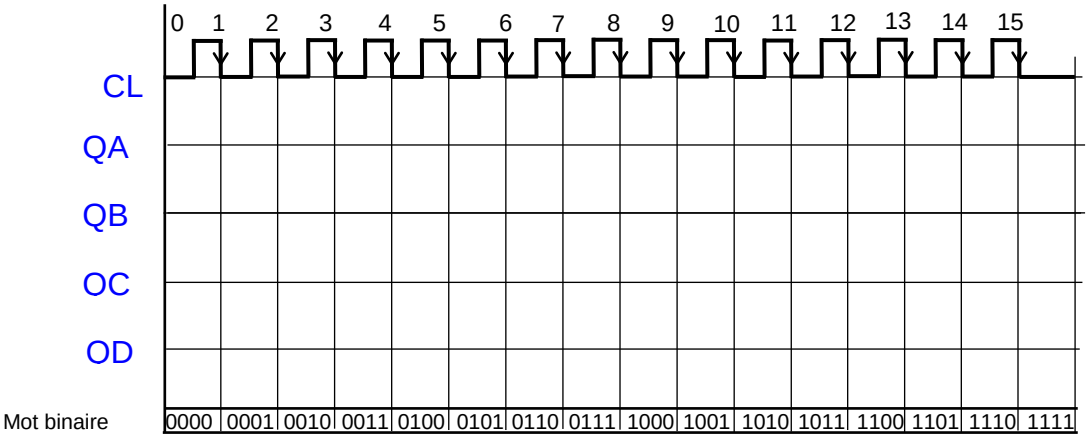


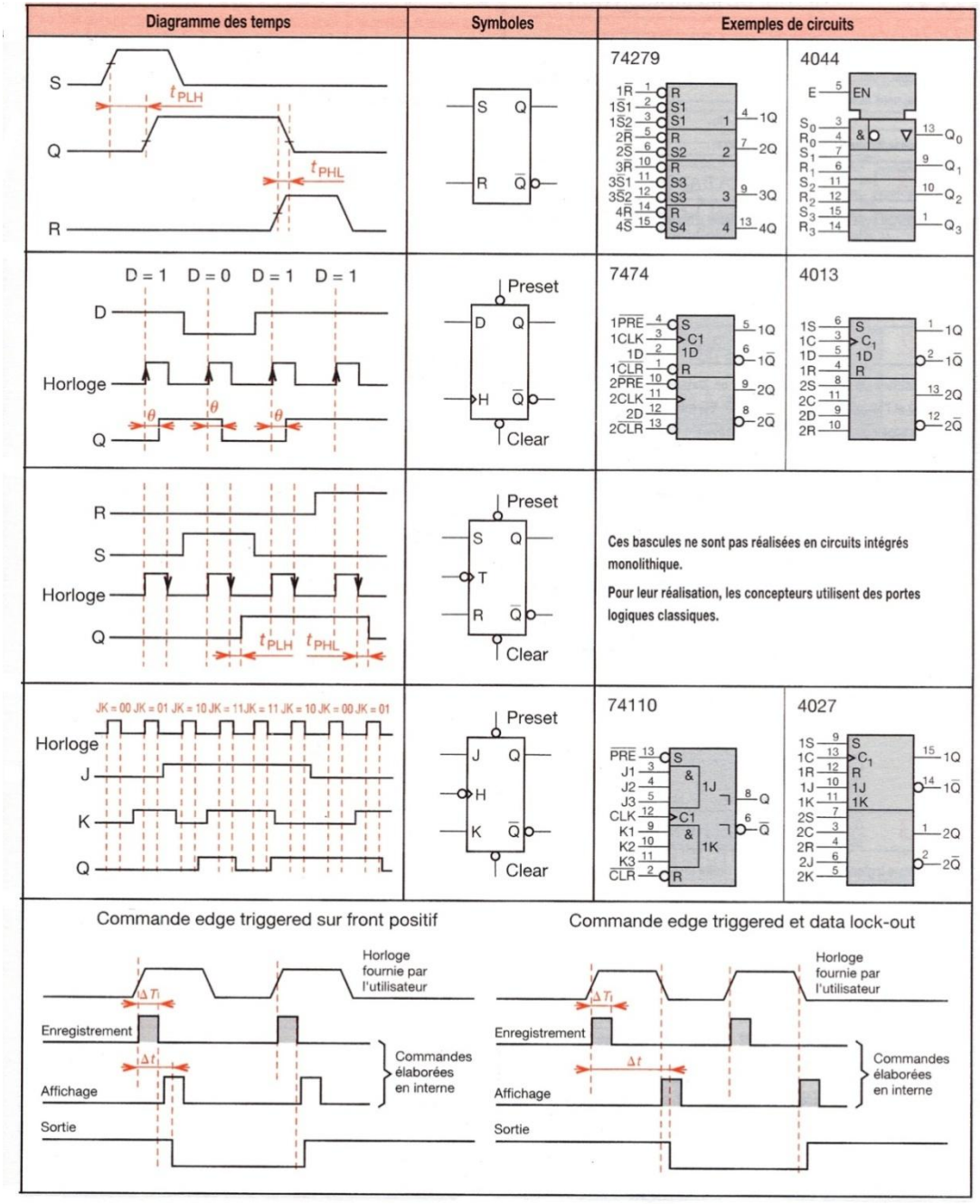
Tableau de fonctionnement

POIDS	2^3	2^2	2^1	2^0
CLK Impulsion	MSB QD	QC	QB	LSB QA
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				

Chaîne d'information

Les compteurs à circuits intégrés disparaissent, car l'évolution des technologies fait que **l'intégration augmente** et le traitement des évènements logiques se fait avec des composants logiques programmables (structure programmable →PAL, FPGA…, séquencement programmable → microcontrôleur, ordinateur, etc…).

LES BASCULES EN CIRCUIT INTEGRE



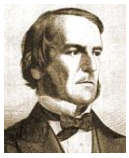
Doc Memotech

Chaîne d'information

Lexique:

Variables logiques : Les variables logiques sont des variables qui ne peuvent prendre que deux valeurs.

Exemple : arrêt marche, ouvert fermé, enclenché déclenché, avant arrière, vrai faux, conduction blocage, allumé éteint, « 0 », « 1 » …..



George Boole : , mathématicien, logicien et un peu philosophe est né le 2 novembre 1815 à Lincoln, dans le Lincolnshire (Angleterre). C'est le père fondateur de la logique moderne. En 1854 il réussit là où Leibniz avait échoué : allier en un même langage mathématiques et symbolisme. Le but : traduire des idées et des concepts en équations, leur appliquer certaines lois et retraduire le résultat en termes logiques. Pour cela, il crée une algèbre binaire n'acceptant que deux valeurs numériques : 0 et 1. L'algèbre booléenne ou algèbre de BOOLE était née. Les travaux théoriques de Boole, trouveront des applications primordiales dans des domaines aussi divers que les systèmes informatiques, les circuits électriques et téléphoniques, l'automatisme…

Claude Shannon : (30avril1916 à Petoskey, Michigan - 24 février 2001 à Medford, Massachusetts) est un ingénieur électricien et mathématicien américain. Il est l'un des pères, si ce n'est le père fondateur, de la théorie de l'information. Son nom est attaché à un célèbre « schéma de Shannon » très utilisé en sciences humaines, qu'il a constamment désavoué.

Augustus de MORGAN : (27 juin 1806 à Madurai (Tamil Nadu) - 18 mars 1871) est un mathématicien et logicien britannique, né en Inde. Il est le fondateur avec Boole de la logique moderne ; il a notamment formulé les lois de De Morgan.

LOGICALC v1.00: petit programme sympathique pour les aficionados de l'automatisme combinatoire. Ce logiciel permet de générer une équation logique simplifiée en partant d'une table de vérité ou d'un tableau de KARNAUGH. Il permet de travailler avec un nombre de variables compris entre 2 et 6.

Maurice KARNAUGH :

Ph.D. de Physique, Université de Yale – 1952, actuellement retraité,
Gouverneur fondateur de l'ICCC en 1972(International Council for Computer Communication)

- Secrétaire général de 73 à 82
- Vice-président de 82 à 86
- Gouverneur émérite depuis

Chercheur aux laboratoires Bell Telephone de 1952 à 1966
Chercheur au centre de recherche IBM – New York – de 1966 à 1993
Professeur d'informatique à l'Institut Polytechnique de New York de 1980 à 1999
Membre de l'IEEE – 1975 – Pour ses travaux sur l'utilisation des techniques numériques en télécommunications