Patrick TRAU
http://pat.fr.st

Les composants de base

de l'automatisme et de l'électronique numérique

Université Louis Pasteur
Institut Professionnel des
Sciences et Technologies
15 rue du Maréchal Lefèbvre
67100 STRASBOURG (F)

Bases d'automatisme − Champ d'application de l'automatisme − vocabul

Table des matières

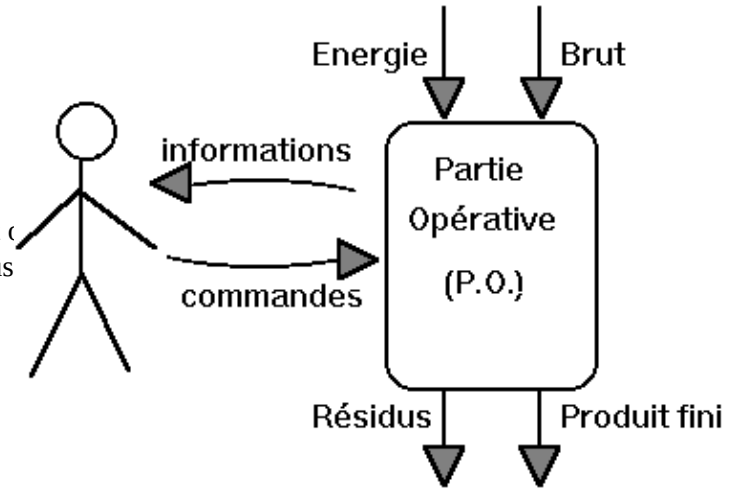
Table des matières

1 Champ d'application de l'automatisme − vocabulaire

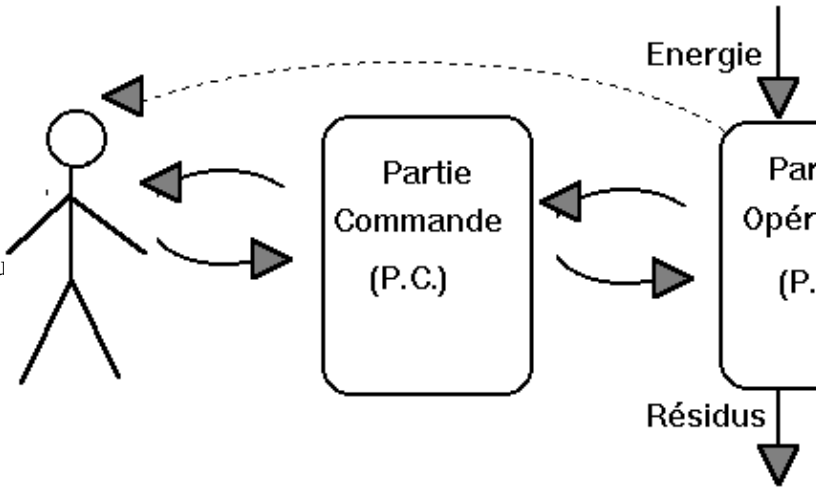
L'automatisme consiste en l'étude de la commande de systèmes industriels.

La première amélioration des conditions de travail a été de remplacer l'énergie humaine fournie par l'ouvrier par une machine (P.O.)

L'opérateur commande la machine, et regarde le résultat c
ses commandes en fonction du déroulement du processus



L'automatisme débute lorsque l'on intercale entre
une P.C. qui prend certaines décisions (gestion au
plus simples et les plus courants).

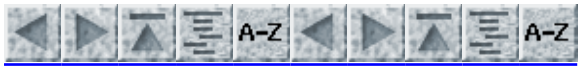


La P.C. lit les informations sur la P.O. par l'intermédiaire de capteurs, et commande les actionneurs de la P.O. Le but est de prendre en compte par la P.C. tout ce qui est répétitif et simple, en laissant à l'opérateur les tâches nobles de réflexion. La P.C. doit nécessairement "tout savoir" : toute information ou commande, même non traitée par elle, devrait passer par elle. Il reste néanmoins quelques phénomènes difficiles à mesurer, ou dont la mesure coûte trop cher par rapport à la probabilité qu'ils se produisent, ou non prévus. Pour cela, l'opérateur−contrôleur reste nécessaire.

Nous nous identifierons toujours à la partie commande. Nous appellerons donc entrées les commandes de l'opérateur ainsi que les informations reçues des capteurs. Nous appellerons sorties les commandes envoyées à la P.O. ainsi que les informations transmises à l'opérateur.

On peut "classifier" les différents cas. La première distinction qui a été faite a été de séparer le Tout Ou Rien (allumé ou non, appuyé ou non, ouvert ou fermé... représenté par 0 ou 1) de l'analogique (grandeurs représentées par une valeur réelle, comme l'électronique par exemple). Désormais, le numérique (gestion de l'analogique par une combinaison de composants ToR, donc regroupement de zéros et de uns pour former des valeurs), tend à englober tous les cas, en premier lieu par son coût plus faible, en second lieu par les possibilités de programmation donc d'évolution.

Un autre distinction peut se faire entre le combinatoire (les sorties dépendent uniquement de l'état actuel des entrées) et le séquentiel (les sorties dépendent des entrées et de l'historique, c'est à dire de ce qui s'est passé auparavant). Le séquentiel en numérique est souvent appelé automatique, en analogique plutôt asservissement.



2 Logique des prédicats

On appelle proposition ou prédicat une "phrase" qui peut être soit vraie, soit fausse. La logique des prédicats est donc un premier exemple de Tout ou Rien (et est utilisée dans les problèmes de reconnaissance de la parole et d'analyse syntaxique).

exemples :

(P1) il pleut
(P2) 6 est supérieur à 4

on notera vrai=1, faux=0 . Donc P2=1, P1 vaut 0 ou 1 suivant les cas.

On peut avoir des propositions dépendant de variables:

X est supérieur à 4
X + Y = 0

On peut également définir des opérateurs : ET (noté . , AND ou ^), OU (+, OR, v) et complément (−− ou /, j'utiliserai / dans ce document car la barre est trop dure à gérer). On peut alors définir pour chacune de ces opérations leur table de vérité qui définit, dans tous les cas, le résultat de l'opération :

P	Q	P.Q
0	0	0
0	1	0
1	0	0
1	1	1

P	Q	P+Q
0	0	0
0	1	1
1	0	1
1	1	1

P	$\bar{P}$
0	1
1	0

On peut également utiliser un tableau à deux entrées pour obtenir un tableau de vérité :

P.Q	0	1
0	0	0
1	0	1

2 colonnes pour les états possibles de Q
2 lignes pour les états possibles de P

En essayant toutes les combinaisons, on peut définir 16 opérateurs binaires (fonctions de deux variables) :

P	Q	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Certains cas sont de peu d'intérêt : a (toujours faux), p (vrai), ou ne dépendent en fait que d'une variable : d (=P), f (=Q), m (=/P), k (=/Q). Les autres ont toutes un nom :

b: ET, h: OU inclusif, o: ON ou NAND, i: NI ou NOR, c: P|Q (P inhibe Q), e: Q|P (Q inhibe P), n: P => Q (implique), l: Q => P (implique), g: OU exclusif (XOR, $\oplus$), j: P <=> Q (équivalent).

3 Algèbre de BOOLE

George Boole (mathématicien anglais, 1815−1864) a démontré que si l'on peut trouver un espace dans lequel certains axiomes se vérifient, alors on se trouve dans un cas singulier, où un certain nombre de théorèmes peuvent s'appliquer.

3.1 axiomes

Pour qu'une algèbre puisse être dite de Boole, elle doit vérifier :

commutativité	a+b=b+a	a.b=b.a
associativité	(a+b)+c=a+(b+c)	(ab)c=a(bc)
distributivité	a(b+c)=ab+ac	a+(bc)=(a+b)(a+c)
éléments neutres	a+0=a	a.1=a
complémentation	$a+\bar{a}=1$	$a.\bar{a}=0$

3.2 théorèmes

Une algèbre de Boole vérifie les théorèmes suivants :

idempotence	a+a=a	aa=a
absorption	a+ab=a	a(a+b)=a
Morgan	$\overline{a+b}=\bar{a}.\bar{b}$	$\overline{a.b}=\bar{a}+\bar{b}$
élément neutre	a+1=1	a.0=0

De plus les fonctions suivantes sont définies :

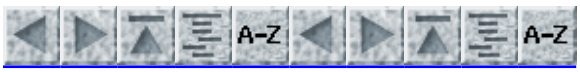
$a \text{ XOR } b = a\bar{b}+b\bar{a}$

$a <=> b = (\bar{a}+\bar{b})(a+b)$

$a => b = a+\bar{b}$

$a \text{ NOR } b = \bar{a}.\bar{b}$

$a \text{ NAND } b = \bar{a}+\bar{b}$



4 Décomposition en NAND − NOR

On peut exprimer toutes les fonctions existantes uniquement à l'aide des fonctions ET, OU et NON (voir exemples précédents). Ceci est pratique dans la mesure où, en automatisme, chaque fonction est représentée par un composant (appelé PORTE). De plus ces trois fonctions correspondent aux fonctions logiques utilisées dans le langage courant. Mais on peut se limiter à 2 fonctions, ET et NON par exemple :

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

On peut même se limiter à une seule fonction (NAND ou NOR) :

$$a.b=\overline{a \text{ NAND } b} \quad ; \quad a+b=\overline{a} \text{ NAND } \overline{b} \quad ; \quad \overline{a}=a \text{ NAND } a$$

On trouvera un exemple au chapitre suivant.

5 Fonctions booléennes à n variables

- tableaux de Karnaugh
- passage ET/OU en NAND

---

ex: $f(a,b,c,d)=\bar{a}bc\bar{d} + \bar{a}b\bar{c} + ab + \bar{a}\bar{b}\bar{c}$

On peut essayer de simplifier son équation à l'aide des relations de l'algèbre de Boole. Mais on peut aussi représenter graphiquement cette fonction par son tableau de vérité :

cd \ ab	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	0	0	1	1
10	0	0	1	1

$f' = \bar{b}c + ac$

Ce tableau nous donne l'état de f (0 ou 1) dans tous les cas (ceci permet de ne pas oublier certains cas particuliers).Dans la pratique on remplit dans le tableau tous les cas où f vaut 1, on complète les zéros dans les cases restantes. On peut remarquer que la fonction f ' est représentée par le même tableau. f ' et f sont donc égales, puisqu'ayant la même valeur dans tous les cas.

5.1 tableaux de Karnaugh

Un tableau de Karnaugh est un tableau de vérité dans lequel les différentes possibilités des entrées sont classées en code GRAY (binaire réfléchi) qui sera détaillé plus loin. Ceci correspond à 0 puis 1 dans le cas d'une variable, 00, 01, 11, 10 pour 2 variables. On peut remarquer qu'en passant d'une case à une case adjacente, une seule variable a changé. Regrouper ces deux cases adjacentes correspond donc à la simplification par cette variable (du fait de l'axiome de complémentation). On simplifie l'équation d'une fonction en faisant des regroupements sous forme d'une, deux ou quatre lignes ou colonnes. Le passage de la dernière ligne à la première est également un cas adjacent (idem pour les colonnes).

En représentant graphiquement une fonction booléenne à n variables, on peut en déterminer une expression simplifiée. Cette simplification est évidente jusqu'à 4 variables, possible avec 5 ou 6 variables (en traitant deux ou quatre tableaux différents supposés superposés). Dans les autres cas, un tableau de Karnaugh permettra toujours de simplifier l'équation, mais jamais au maximum. Par exemple :

$f=\bar{a}.b.c + \bar{a}.b.\bar{c} + \bar{a}.\bar{b}.\bar{c} + b.c + a.c$

c \ ab	00	01	11	10
0	1	1	0	0
1	0	0	1	1

donc : $f=a.c + \bar{a}.\bar{c} + \bar{a}.b$ ou $f=a.c + \bar{a}.\bar{c} + b.c$

Remarque : Dans la pratique, si certains cas sont indifférents (par exemple combinaison de capteurs impossible), on place un X dans le tableau, que l'on mettra à 0 ou 1 pour simplifier l'expression finale.

On pouvait également regrouper les 0 (s'il y en a moins ou plus groupés) :

f=(a b c)+(a c) d'où f=(a+b+c)(a+c)=ac+ba+bc+a c

On peut également développer la fonction en somme de MINTERMES (forme canonique disjonctive).

Dans notre cas : f=a.b.c+a.b.c+a.b.c+a.b.c+a.b.c

Propriétés : Il y a 2^n mintermes d'ordre n (n étant le nombre de variables de la fonction). Deux mintermes de même ordre ont un produit nul. La somme de tous les mintermes d'ordre n est 1.

L'intérêt de ces décompositions est leur unicité, donc en particulier une programmation facilitée.

exercices : trouver la forme réduite et les formes canoniques des fonctions :

a) $y = c.\overline{d} + \overline{a}.\overline{b}.\overline{c} + a.\overline{b}.c + b.c + \overline{a}.\overline{b}c.d$

b) $z = \overline{(\overline{a}+b)\overline{c}}.(\overline{b}+db) + \overline{abc}$

question : où cliquer pour la solution ?

5.2 passage ET/OU en NAND

on notera $\overline{abc}$ par NAND[a,b,c]

On décompose la fonction sous forme canonique : (comme vous pouvez le voir, je trouve que j'ai assez travaillé, je garde mes équations en mode texte bien que ce soit moins beau)

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

on peut alors écrire :

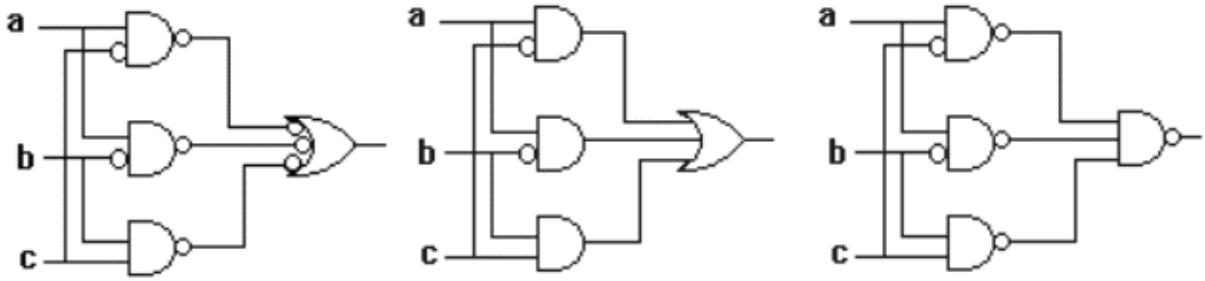
$$f = (a\overline{b}c) + (ab\overline{c}) + (\overline{a}b\overline{c})$$

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

$$f = \mathrm{NAND}[\,(a,\overline{b},c)\ ,\ (a,b,\overline{c})\ ,\ (\overline{a},b,\overline{c})\,]$$

$$f = \mathrm{NAND}[\,\mathrm{NAND}(a,\overline{b},c)\ ,\ \mathrm{NAND}(a,b,\overline{c})\ ,\ \mathrm{NAND}(\overline{a},b,\overline{c})\,]$$

Dans la pratique, les schémas de fonctions définies par ET / OU se transforment facilement en réseaux composés uniquement de NAND. Il suffit d'inverser toutes les extrémités des liaisons internes :



La symbolisation est détaillée dans le transparent 5

6 Applications de l'algèbre de BOOLE

6.1 logique des prédicats

propositions, vrai, faux, et, ou, non...

exercice : calculez : NON(Pat est beau ou Pat est intelligent)

6.2 ensembles

intersection ^, union U, complément

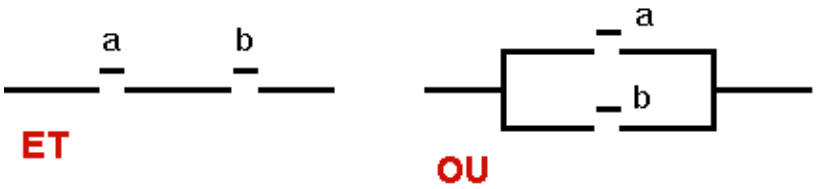
ex: absorption: $A \cap (A \cup B) = A$

regardez le transparent 1B

6.3 circuits électriques

0 représente un interrupteur (je devrais dire en toute rigueur contacteur) ouvert (le courant ne passe pas)
1 représente un interrupteur fermé (le courant passe)
Un contacteur normal laisse passer le courant quand on l'actionne, un contacteur inverse quand on le laisse au repos.

On effectue une fonction ET par la liaison de 2 interrupteurs (il faut appuyer sur a ET b pour que le courant passe) en série (il faut appuyer sur a ET b pour que le courant passe), la fonction OU par la liaison parallèle (il faut appuyer sur a OU b pour que le courant passe).



On peut vérifier les axiomes de l'algèbre de Boole en représentant les différents schémas (transparent T1).

On pourrait de même vérifier ce que donnent les théorèmes (inutile de les démontrer puisque les axiomes sont vérifiés).

Applications : Soient un moteur M fonctionnant en 220 V, un relais électromagnétique R comportant le contact inverseur, 3 boutons contacts A B et C fonctionnant sous ambiance humide (tension de commande 12 V). Le moteur doit fonctionner si l'on appuie sur A et B simultanément. Toute action sur C (arrêt d'urgence bistable) doit arrêter le moteur. (solution R = a.b.c, M = R). Par mesure de sécurité (centrale nucléaire par exemple), on triple les capteurs. Ces trois capteurs (mesurant la même chose) commandent un relais. On veut

que, en cas de défaillance d'un capteur, le relais continue à fonctionner mais qu'une des 3 lampes L1, L2, L3 indique le capteur défaillant. Les 3 capteurs sont appelés a,b,c, le relais R.

$$R = abc + ab\bar{c} + a\bar{b}c + \bar{a}bc$$

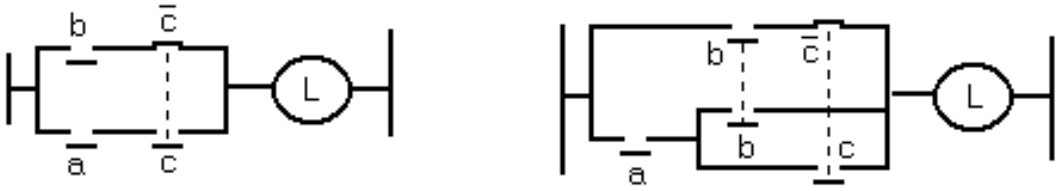
$$L1= \bar{a}bc + a\,\bar{b}\,\bar{c} \;\; ; \;\; L2= a\bar{b}c + \bar{a}bc \;\; ; \; L3= ab\bar{c} + \bar{a}cb$$

6.4 aléas technologiques

Réalisons la fonction L = b.(/c) + ac en faisant le schéma électrique. On peut remarquer que si a=1 et b=1, par passage par exemple de c=1 à c=0, il y a un instant où L est coupé (entre le contact c et le contact inverse). C'est l'aléa technologique. Pour y remédier, il faut faire des recouvrements dans le tableau de Karnaugh. On obtient :

$$L = b\bar{c} + ac + ab$$

On peut vérifier sur le schéma électrique que l'aléa a disparu. En fait on a un aléa quand pour un passage d'une case valant 1 à une case adjacente valant également 1 (un capteur a changé), on traverse une discontinuité.



6.5 les circuits pneumatiques

<u>première convention :</u>

- 1= conduite libre
- 0= conduite fermée

ce problème est analogue aux circuits électriques, et résout les "problèmes de robinets". On utilise, à la place de l'interrupteur, le distributeur (bistable ou monostable). Le ET se fait par montage en série, le OU en parallèle. (T2)

<u>seconde convention (plus courante) :</u>

- 1= pression P (+ ou − epsilon)
- 0= pression p (p<P)

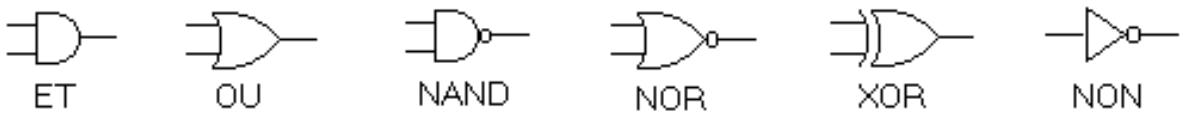
On utilise des distributeurs à commande pneumatique (équivalents à des relais).Le ET se fait par montage série, le OU en parallèle, avec adjonction d'un clapet anti−retour pour que la pression de b ne sorte pas par l'échappement de a.

Remarque (voir T2) : pour le OU : en faisant le tableau de Karnaugh de la fonction OU, on peut remarquer que f = a + b correspond à un recouvrement, alors que f'= a + ba est équivalente, mais sans recouvrement. En effectuant le schéma du OU sans recouvrement, on remarque que la cellule anti−retour est inutile. En fait, chaque recouvrement entraîne l'adjonction d'une cellule anti−retour. Contrairement à l'électrique, il faut donc supprimer les recouvrements.s.

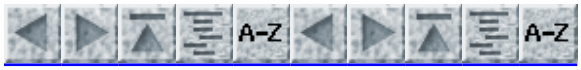
6.6 l'électronique (portes)

On utilise, comme en pneumatique seconde convention, un niveau 1 (5V par ex), et un niveau 0 (0V). Les fonctions sont effectuées à l'aide de "portes", (circuits intégrés). On trouve couramment les inverseurs, NOR et NAND, mais aussi ET et OU. Les portes comportent entrée(s) et sortie, le courant ne pouvant pas "remonter" au travers d'une porte, ce qui évite le problème des anti−retour (effectués, au cas où, à l'aide d'une diode). De toute façon, le courant ne transite pas par les portes. Chacune est alimentée. Si sa sortie doit être à 1, la sortie est commutée sur l'alimentation. Donc même si le signal d'entrée n'est pas propre (mais reste dans l'intervalle de tolérance prévu, au dessus de 2,7V en TTL), le signal de sortie sera propre.

Représentation des portes : voir transparent T5. J'utiliserai principalement la norme MILSTD :



Il faut remarquer que du fait du faible coût de ces composants, il revient généralement moins cher de réaliser une partie commande électronique, et d'utiliser des interfaces appropriés pour commander les actionneurs. Ce n'est cependant pas vrai dans deux cas : si le problème est très simple (pour allumer la lumière de mon garage je mets directement les interrupteurs en parallèle, pas besoin d'une porte OU et d'un interfaçage 220V − TTL, encore moins d'un automate programmable en Grafcet) ou en cas d'environnement incompatible (humide, parasité...).



7 Combinatoire numérique

Nous nous limiterons aux applications électroniques (les seules existantes, mis à part l'ordinateur à eau du musée du CNAM).

7.1 Représentation des nombres entiers

7.1.1 la base 2

On peut représenter des nombres par une combinaison de zéros et de uns. Chaque chiffre binaire (BInary digIT) est appelé BIT. 8 bits forment un octet (BYTE). Mais plusieurs codifications sont envisageables. La plus utilisée est le binaire (ou binaire naturel en cas d'ambiguïté).

passage binaire (indice b) −> décimal (indice d) :

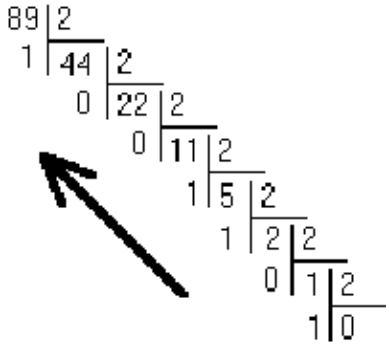
1011001_b représente :

$1x2^6 +0x2^5 +1x2^4 +1x2^3 +0x2^2 +0x2^1 +1x2^0$

$=1x64 +0x32 +1x16 +1x8 +0x4 +0x2 +1x1$

$= 89_d$

à l'inverse, pour transformer 89_d en binaire, on peut utiliser la méthode des divisions successives par 2 : on divise successivement par 2 jusqu'à un résultat de 0, les restes successifs (de bas en haut) forment le nombre binaire.



De tête, je ferai : 89 = **1**x64 reste 25 donc **0**x32, **1**x16 reste 9, **1**x8 reste 1 donc **0**x4, **0**x2, **1**x1.

7.1.2 la base 16 (hexadécimal)

On utilise les chiffres 0 à 9 puis les lettres A à F. $3A5_h$ vaut $3x16^2 + 10x16^1 + 5x16^0 = 3x256 + 160 + 5 = 933_d$. On passe de l'hexa au décimal par divisions successives par 16. Transformer de l'hexadécimal en binaire est enfantin : il suffit de remplacer chaque chiffre par sa valeur binaire sur quatre bits : $3A5_h = 0011\ 1010\ 0101_b$ (on peut vérifier que ça vaut 933_d). En effet, 001110100101_b
$= 0.2^{11} +0.2^{10} +1.2^9 +1.2^8 +1.2^7 +0.2^6 +1.2^5 +0.2^4 +0.2^3 +1.2^2 +0.2^1 +1.2^0$
$= (0.2^3 +0.2^2 +1.2^1 +1.2^0)2^8 +(1.2^3 +0.2^2 +1.2^1 +0.2^0)2^4 +0.2^3 +1.2^2 +0.2^1 +1.2^0$
$= 0011_b x2^8 +1010_b x2^4 +0101_b x2^0$
$= 3_d x16^2+10_d x16+5_d$
$=3A5_h$

Contrairement à ce que beaucoup de gens croient, aucune machine ne compte en hexadécimal. Elles travaillent toutes en binaire, et ne se servent de l'hexa que pour dialoguer avec nous (nous nous trompons trop souvent dans de longues listes de 0 et 1).

7.1.3 le Décimal Codé en Binaire (DCB ou BCD en anglais)

Si vous achetez un voltmètre numérique (69F en supermarché), la valeur mesurée est transmise à l'afficheur en numérique. Mais elle est auparavant transformée en décimal, chaque chiffre décimal est transmis à un afficheur en binaire naturel (sur 4 bits). C'est le BCD : la juxtaposition des valeurs binaires (sur quatre bits) des chiffres décimaux. Donc 583_d se notera $0101\ 1000\ 0011_{bcd}$. Cette codification pose deux problèmes principaux :

- un certain nombre de combinaisons ne sont pas utilisées (celles qui correspondent à A à F en hexa). Sur 8 bits on représente les nombres de 0 à 99 au lieu de 0 à 255 en binaire. Sur 16 bits, on se limite à 9999 au lieu de 65535...
- les calculs sont compliqués : rien que pour faire un programme d'incrémentation (ajouter 1), il faut ajouter 1 aux 4 bits de droite, si on obtient 1010 (10_d), on les remplace par 0000 et on ajoute 1 aux quatre bits suivants (dizaines), sans oublier de vérifier si l'on ne passe pas à la centaine suivante...

Dès qu'il y a des calculs à effectuer, les systèmes numériques traduisent les nombres BCD en binaire dès leur acquisition, les résultats seront transformés en BCD au moment de leur sortie. On peut remarquer que pour transformer un nombre binaire en décimal (bcd), l'ordinateur est obligé de faire des divisions successives par 1010 (10 en binaire)

7.1.4 le binaire réfléchi (code GRAY)

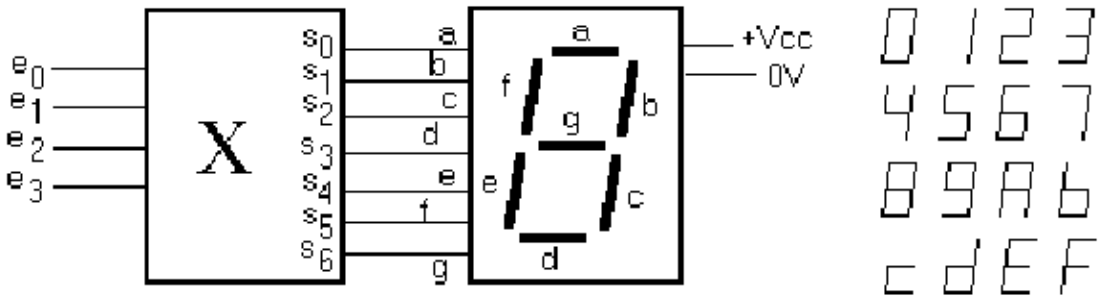
On désire qu'en passant d'un nombre à son suivant (+1) ou précédent (−1), on n'aie qu'un seul bit qui change. On désire de plus que les zéro rajoutés à gauche d'un nombre ne soient pas significatifs. Sur deux bits, on utilisera les codes 00, 01, 11 puis 10. Sur 3 bits, on gardera les mêmes premiers codes (précédés d'un zéro). La combinaison suivante débutera donc obligatoirement par 1, donc les deux autres bits ne peuvent pas changer. On continuera à prendre les mêmes codes, en ordre inverse, débutant par 1 : 110, 111, 101 et 100. En passant à 4 bits, on précède ces 8 cas d'un 0, les 8 suivants étant les mêmes, dans l'ordre inverse, précédés d'un 1. Ce codage est utilisé dans les cas où des valeurs ne peuvent varier que par incrémentation ou décrémentation : si l'on voit que plus d'un bit a changé entre deux valeurs, c'est qu'il y a eu un problème (en général le nombre a changé trop vite, le système n'a pas eu le temps de lire toutes les valeurs). Il faut par contre passer en binaire naturel pour tout autre calcul que l'incrémentation.

Un exemple est le capteur de position angulaire (voir transparent T4). Un capteur incrémental comptant des impulsions est utilisé par exemple sur les robots. C'est un disque, entaillé d'encoches régulièrement espacées, passant devant un capteur optique. Certaines impulsions trop rapprochées peuvent être "oubliées" en cas de choc par exemple, et donc occasionner un mauvais réglage. A l'initialisation et en cas de problème, on doit ramener toutes les articulations en position de repos, puis mettre les compteurs à 0, avant de pouvoir utiliser le robot. Un capteur absolu quand à lui donne toujours le position exacte (bien qu'il y ait souvent une

démultiplication, le mouvement total fait plus d'un tour mais aucun choc ne fera sauter le capteur d'exactement un tour). On utilise un code binaire réfléchi car un autre codage nécessiterait, pour passer d'une valeur à la suivante, une modification simultanée de plusieurs bits (voir explication sur T4)

7.2 Applications

7.2.1 l'afficheur 7 segments



Le segment a s'allume si s_0=0, s'éteint sinon (idem pour les autres segments). On cherche le schéma interne du composant X qui permet d'afficher le chiffre correspondant au nombre binaire disponible en entrée (e_0 bit de poids faible) On peut en premier lieu faire une table de vérité, donnant l'état des 7 sorties s_i pour les 16 combinaisons possibles des 4 entrées e_i. Puis on peut rechercher l'équation de chaque sortie en fonction des entrées (Karnaugh par exemple), puis on peut rechercher si certains termes apparaissent dans plusieurs équations afin de ne pas câbler plusieurs fois la même fonction.

En utilisant plusieurs afficheurs, on affichera un nombre binaire en hexa ou un nombre BCD en décimal, pour des nombres de bits supérieurs à 4

7.2.2 l'additionneur binaire

étudier le circuit : (transparent T3)

$s = r \text{ NOR } 2 \;;\; 2 = a \text{ NOR } b = \bar{a}.\bar{b} \;;\; r = 3 \text{ NOR } 4 \;;\; 3 = b \text{ NOR } b = \bar{b} \;;\; 4 = \bar{a}$

	a	b	r	s
	0	0	0	0
	0	1	0	1
	1	0	0	1
	1	1	1	0

$\text{donc } r = ba \;;\; s = \bar{b}a + \bar{a}b$

c'est donc un additionneur binaire. (s=somme r=retenue, 0+0=0, 1+0=1, 1+1=0 et je retiens 1).

Exercice : faire le même composant uniquement avec des NAND.

A l'aide de ce composant, on peut maintenant faire un additionneur sur plusieurs bits (en général au moins 8, analysons ici uniquement le cas de 3 bits).

$a_2\ a_1\ a_0$

$+\ b_2\ b_1\ b_0$ −> r0, r1, r2 sont les retenues intermédiaires

$r_3\ s_2\ s_1\ s_0$

s0=a0+b0 (retenue éventuelle s0). s1=s0+a1+b1 (retenue éventuelle s1, qui ne peut valoir que 0 ou 1, donc la retenue ne peut provenir exclusivement que d'une seule des deux additions). On répète ensuite le même principe, pour tous les bits désirés

7.2.3 décodeur binaire −> code Gray (T4)

Trouver le schéma d'un composant admettant en entrée un nombre binaire naturel, donnant en sortie son équivalent en code binaire réfléchi. On se limitera aux nombres de 3 bits. On peut trouver g1 et g0 par tableau de Karnaugh:

```
---+------+------+
   |b2b1b0|g2g1g0|
---+------+------|
 0 | 0 0 0| 0 0 0|
 1 | 0 0 1| 0 0 1|  g2 = b2
 2 | 0 1 0| 0 1 1|
 3 | 0 1 1| 0 1 0|  g1 = b2.b1 + b2.b1 = b1 XOR b2
 4 | 1 0 0| 1 1 0|
 5 | 1 0 1| 1 1 1|  g0 = b1 b0 + b0 b1 = b0 XOR b1
 6 | 1 1 0| 1 0 1|
 7 | 1 1 1| 1 0 0|
-----------------+
```

7.2.4 décodeur 3/8, encodeur, multiplexeur, démultiplexeur (T6)

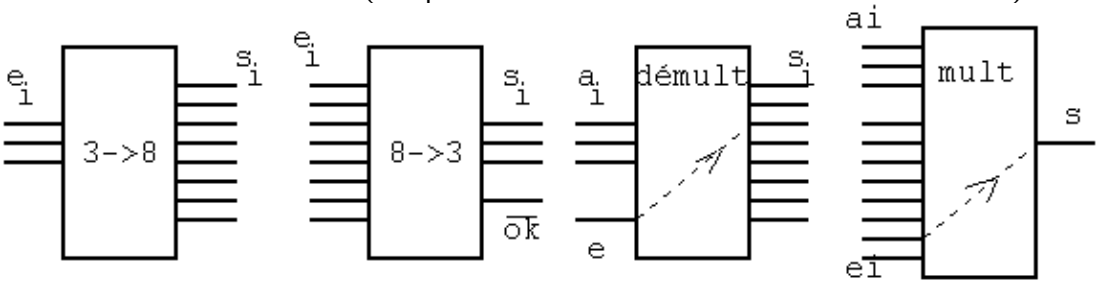
Le décodeur 3 dans 8 comporte 3 entrées e0 à e2, 8 sorties s0 à s7. Il allume une et une seule sortie à la fois, celle correspondant à la valeur binaire donnée en entrée (dons entre 0 et 7, ce qui fait entre 000 et 111 en binaire). A l'inverse, un encodeur 8 dans 3 pose quelques problèmes : quelle sortie donner si plusieurs entrées sont allumées ? On prévoit en général soit une priorité, soit une sortie supplémentaire signalant l'erreur.

Le multiplexeur (voir transparent T6) comporte 2n (ici 8) entrées d'information

- n (ici 3) entrées de sélection (entrée d'un nombre I en binaire)
- 1 sortie (recopie de l'entrée d'information (numéro I) sur la sortie)

Le démultiplexeur comporte 2n (ici 8) sorties

- n (ici 3) entrées de sélection (entrée d'un nombre I en binaire)
- 1 entrée d'information (recopie de l'entrée d'information sur la sortie numéro I)

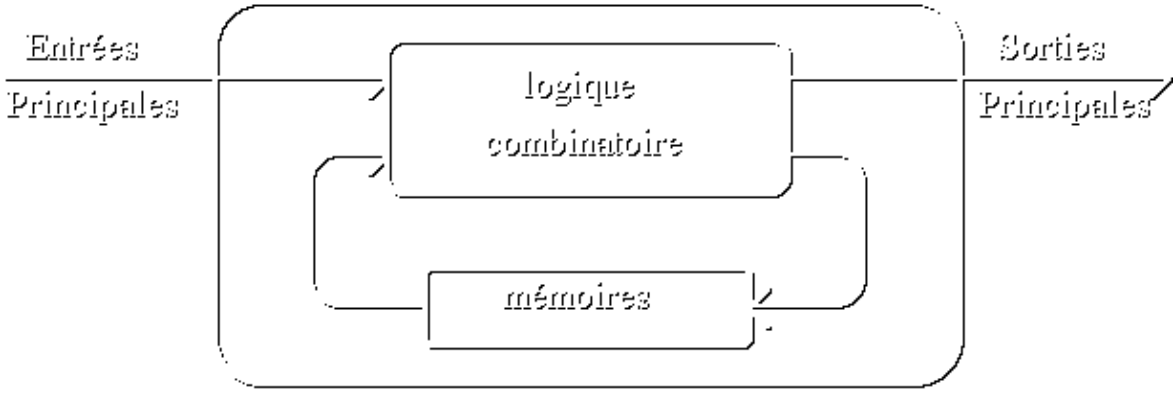


(j'ai représenté toutes les entrées à gauche, les sorties à droite)

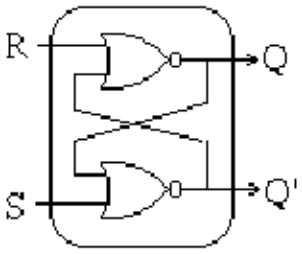
8 Séquentiel (câblé)

8.1 Définition

En combinatoire, pour chaque combinaison des entrées, il existe une et une seule combinaison des sorties. En séquentiel, l'état des sorties dépend en plus de l'histoire (de l'état précédent, qui lui aussi, dépend de l'état qui l'a précédé...)



8.2 bascule R S



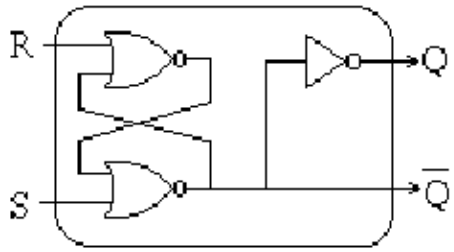
Ce composant est le composant de base du séquentiel. Analysons l'état des sorties dans différents états.

Dans cette table de vérité, on considérera un déroulement séquentiel : les combinaisons des entrées se suivent dans le même ordre

Rappel : la sortie d'une porte NOR ne vaut 1 que quand toutes ses entrées sont à 0.

S	R	Q	Q'	
1	0	1	0	Set (allumer)
0	0	1	0	Mémorisation
0	1	0	1	Reset (éteindre)
0	0	0	1	Mémorisation
1	1	0	0	Etat Interdit !

Si le dernier cas n'est pas utilisé (on ne demande pas simultanément d'allumer et d'éteindre), Q' vaut toujours l'opposé de Q, on l'appellera donc /Q (Q barre)



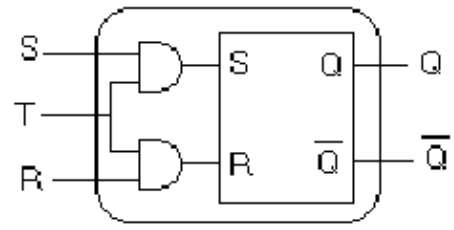
On réalise ici une bascule à enclenchement prioritaire (idem, excepté si S=R=1, Q et mis à 1). Ici, dans tous les cas, Q' est l'opposé de Q. Un bascule à priorité déclenchement aura également le même comportement qu'une RS, excepté dans l'état interdit où /Q vaudra 1

Application : **circuit anti rebond**

Un capteur ne peut pas passer de manière parfaite (sans aléa) de l'état 0 à l'état 1. On peut utiliser une bascule, qui mémorisera l'état stable précédent pendant l'état transitoire. (T1)

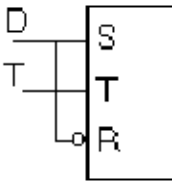
Exercice : quel est le comportement d'un même composant où l'on a remplacé les NOR par des NAND ? (réponse : bascule commandée par des niveaux 0)

8.3 bascule RST



T est l'entrée de validation : si T=1, les entrées R et S sont prises en compte, si T=0 elles ne le sont pas. Dans ce cas, la bascule n'est pas éteinte, elle reste "figée" dans le même état.
Souvent, la bascule comporte deux entrées supplémentaires : Preset (forçage à 1, quel que soit l'état de T) et Clear (forçage à 0), qui permettent de forcer la bascule même si T=0, utilisées généralement pour l'initialisation du composant.

Application : **bascule D** ou Latch ou mémoire : on possède une entrée D, reliée à S d'une RST, et à R par l'intermédiaire d'un inverseur. La sortie Q vaudra l'état lu et mémorisé lors du dernier T=1. C'est le composant de base d'une mémoire d'ordinateur : est mis à 1 ou 0 au moment voulu, figé le reste du temps.

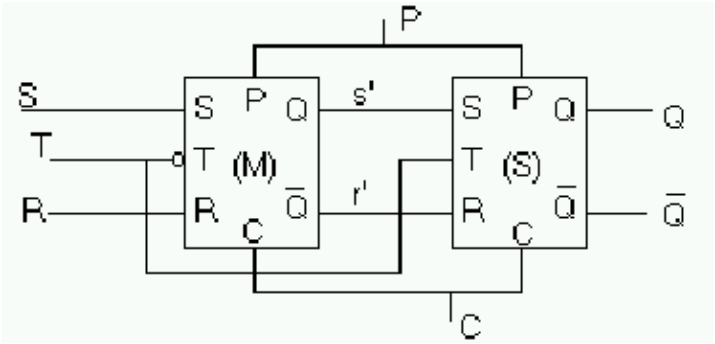


8.4 maître esclave

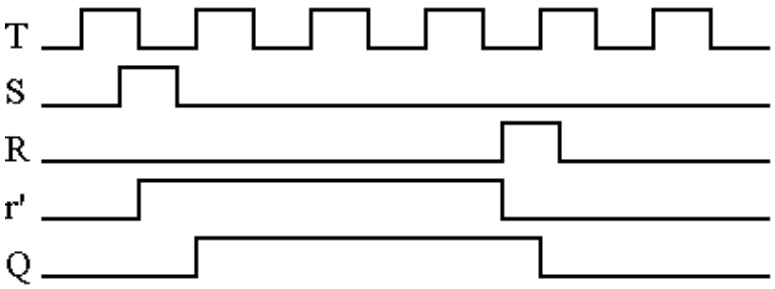
8.4.1 fonctionnement

Deux bascules RST sont reliées en série. Une seule est validée à la fois (T inversé). Une entrée Preset permet le forçage à 1 de l'ensemble, une entrée Clear le forçage à 0 (indépendamment de l'état précédent et de T).

Analysons le fonctionnement de cette bascule:



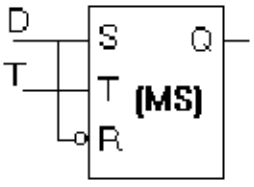
	maître (Master)	esclave (Slave)
Si T=0	information S/R transmise en s'r'	non transmis en Q (ancien Q)
Si T=1	R S en attente (ancien r's')	ancien r's' transmis en Q



On remarque donc que l'information est transmise au prochain front montant de l'horloge T.

8.4.2 bascule D MS

Elle synchronise un signal extérieur sur un front d'horloge (à condition que le signal dure au moins une demi période).

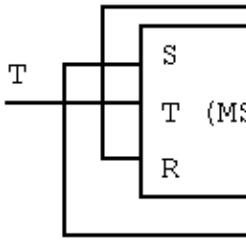


8.4.3 cas particulier : la bascule JK

Elle comporte en général plusieurs entrées de mise à 1 (J) et de mise à 0 (K). Dans le cas où l'on demande l'état impossible (J.K), la sortie est inversée à chaque front d'horloge. Dans les autres cas le fonctionnement est identique. De plus on peut avoir des bascules JK dont le basculement et commandé par un niveau (0 ou 1) ou par un front ("edge triggered").

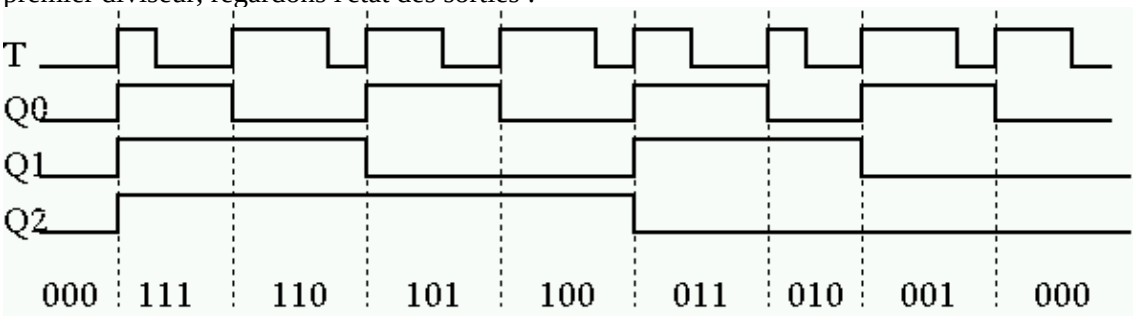
8.4.4 le diviseur de fréquence

A chaque front montant de T, la sortie Q change d'état. Si le signal T est carré, de fréquence F, alors Q sera carré de fréquence F/2. En disposant en série plusieurs diviseurs en cascade, on obtient un compteur ou décompteur binaire (même si T n'est pas régulier) :



8.4.5 le compteur − décompteur

On dispose donc en série des diviseurs de fréquence (trois par exemple), à chaque front appliqué en entrée du premier diviseur, regardons l'état des sorties :

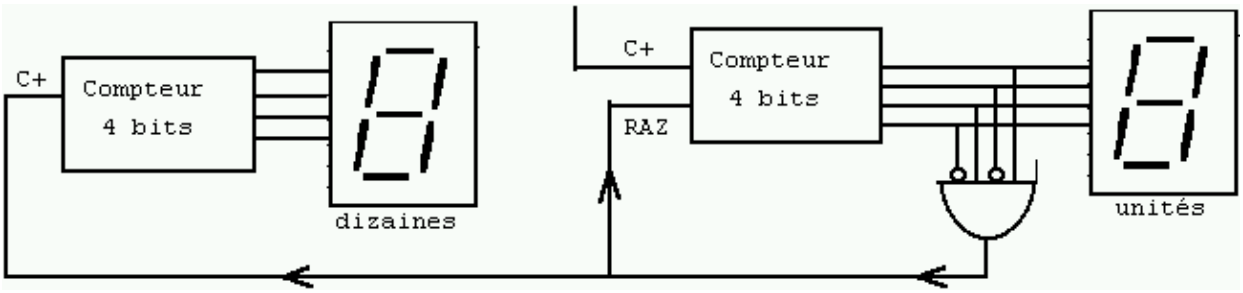


Ce sytème décompte les fronts (même si les signaux d'entrée ne sont pas régulièrement espacés). En inversant les sorties, on obtient un compteur. En reliant ensemble les Clear des différents étages, on peut remettre le compteur d'impulsions à 0. En général, on initialise le décompteur (par les P et C) au nombre à compter, et on attend la valeur 0. On peut remarque le binaire se "crée" automatiquement : la base 2 est la mieux adaptée au comptage à l'aide de composants ToR.

Un compteur − décompteur comporte deux entrées, une de comptage (ajoute un) et l'autre de décomptage (soustrait un). Un compteur asynchrone (comme celui−ci) pose un petit problème : les bascules en série ont un temps de réponse qui fait que la nouvelle valeur se "propage" de gauche à droite, on aura donc pendant un très court instant une valeur de sortie erronnée. Les compteurs synchrones résolvent ce problème (à l'aide de bascules JK, je ne donne pas le schéma, il faut bien que les éditeurs de livres scolaires aient encore quelque chose à vendre).

8.4.6 compteur BCD

Une fois arrivé à la valeur 1010 (10 en binaire), on le remet à 0 et on lance un signal à la dizaine supérieure. Attention dans la pratique ce n'est pas aussi simple : si un composant va plus vite que l'autre, la remise à 0 peut se faire sur une période transitoire où l'on se trouvait à 1010. (C+=entrée de comptage)

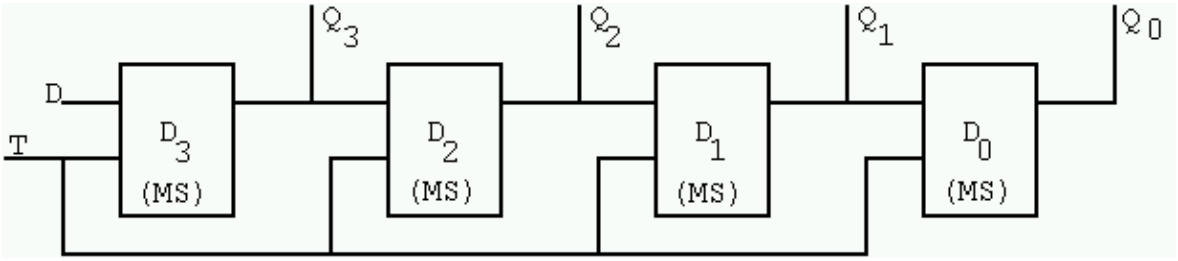


8.4.7 le fréquencemètre

Autre application du compteur. C'est un compteur d'impulsions pendant un temps donné (cas des fréquences élevées), ou alors on compte le temps pendant une période (fréquences faibles).

8.4.8 le registre à décalage

De même en mettant en cascade des bascules D MS, on obtient un registre à décalage, que l'on peut initialiser en parallèle (indépendamment de T) par les broches Preset et Clear.



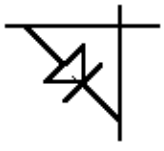
à chaque front de T la valeur Q_i est décalée en Q_{i-1}. Pour un bon fonctionnement, il faut que chaque D_i soit déclenché <u>après</u> D_{i-1}.

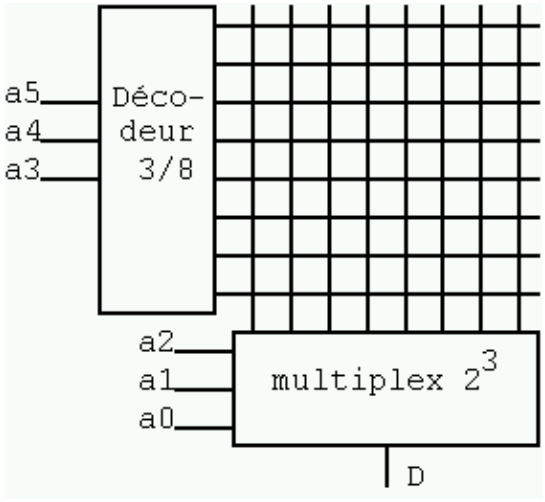
8.5 mémoires

8.5.1 principe

ROM : Read Only Memory : On a figé par construction le contenu des mémoires. En fait, pour un bit, une mémoire à 1 correspond à une liaison sur l'alimentation, un 0 à une liaison à la masse. Comment regrouper plusieurs bits ? Soit par exemple une mémoire de 64 valeurs binaires :

Les intersections sont : pour une valeur 0 : pas de liaison, pour une valeur 1 : une diode empêchant le courant de remonter de la colonne. :





En entrant une ADRESSE (numéro de mémoire, entre 0 et 63 ici) sous forme binaire, on obtient la donnée désirée (contenu de la mémoire). L'adresse se décompose en une partie haute (a3 à a5) déterminant la ligne mise à 1, et une partie basse (a0 à a2) déterminant la colonne connectée sur la sortie D

PROM : programmable une seule fois : liaison "fusible". On programme une Prom sous une tension supérieure à la tension de fonctionnement.

UVPROM : Prom reprogrammable après régénération sous ultra violets (20 mn).

EEPROM : Prom régénérable électriquement.

Application :Utilisation de ROM en combinatoire

Pour chaque état des entrées on mémorise la sortie (décomposition en mintermes). Par exemple, pour créer un générateur de caractères 8x8 pixels on préférera utiliser une ROM plutôt que de faire un circuit spécifique (pour 127 caractères, il faut 1 Ko).

RAM statique : garde la valeur tant qu'elle est alimentée. On utilise la même disposition que pour la ROM, mais à chaque intersection on place une bascule.

RAM dynamique : d'accès beaucoup plus rapide, mais il faut les régénérer (lire et réécrire) à intervalle régulier (plusieurs fois par seconde). On intègre actuellement plusieurs Mbit par composant.

8.5.2 brochage

Un boîtier RAM comprend en général des entrées A_0 à A_n permettant de désigner la mémoire, R/W pour dire si lire ou écrire, et D_0 à D_m pour les données (entrée sortie) (ou D si c'est un boîtier de mémoires 1 bit). De plus, le composant ne fonctionnera que s'il est sélectionné (entrée CS : chip select). De plus, il faut entrer un signal de synchronisation (horloge) et évidement l'alimenter.

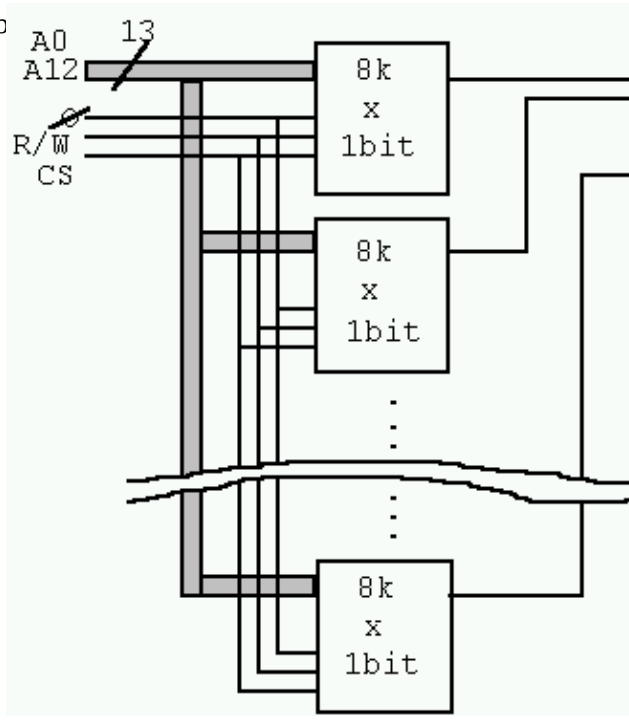
Chronogramme : en lecture, il faut donner l'adresse, CS, Read, on obtiendra le contenu D au prochain top d'horloge. En écriture, on donne l'adresse, CS et Write, puis la donnée au prochain top d'horloge.

8.5.3 association de boîtiers mémoire

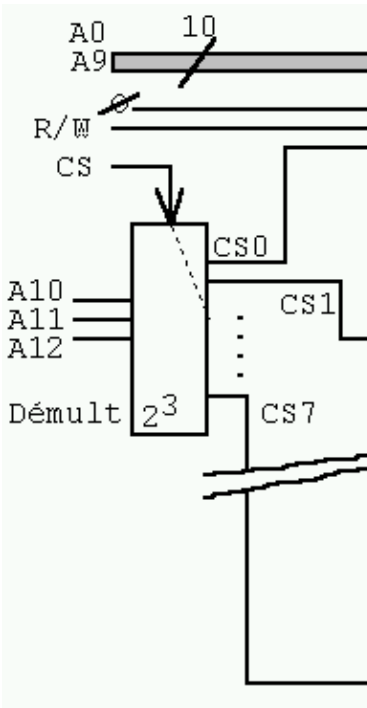
Comment utiliser 8 boîtiers de 8k x 1 bit pour créer une mémoire de 8k x 8 bits ?

En fait, on envoie les signaux de commande et l'adresse aux 8 boitiers. Ceux−ci,

simultanément, traiteront les 8 bits du mot désiré. Les différents b
mémoire ne sont donc pas physiquement situés au même endroit.



Et avec 8 boîtiers de 1k x 8 bits ?

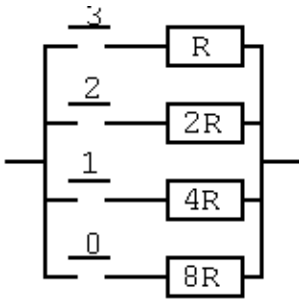


Ici, les 10 bits de poids faible de l'adresse désirée est transmise à tous les boitiers. Mais un seul est sélectionné, suivant les 3 bits de poids fort de l'adresse. Les 8 bits de données de tous les boitiers sont reliés ensemble, on est sûr qu'un seul sera sélectionné à la fois, via le démultiplexeur.

9 conversion numérique analogique

Remarque : Ce paragraphe n'est plus traité en IUP1

9.1 conversion numérique analogique (CNA)



On désire transformer une valeur numérique (en binaire) discrète en une tension variable.

On utilise un ampli opérationnel (la tension de sortie est proportionnelle à la résistance d'entrée). On lui applique à l'entrée le schéma ci−contre:

On obtient une résistance équivalente, en fonction des 4 "contacteurs" telle que définie ci−après :

3	2	1	0	R_{eq}/R
0	0	0	0	0/8
0	0	0	1	1/8
0	0	1	0	2/8
0	0	1	1	3/8
0	1	0	0	4/8
...	...	...	...	...
1	1	1	0	14/8
1	1	1	1	15/8

Pour 16 bits de précision, on arrive à une résistance de 32768 R. Pour que la valeur du dernier bit soit inférieure à l'erreur, il faut une précision de plus de 1/32768 sur les résistances (dans la plage de températures d'utilisation).

Pour résoudre ce problème, on utilise le montage R/2R (T3). A chaque intersection, le courant est partagé en deux (résistance équivalente égale).

9.2 conversion analogique numérique (CAN)

Une solution rapide est de comparer en parallèle la tension à mesurer avec l'ensemble des possibilités. Mais la précision obtenue est faible et la mise en oeuvre complexe.

Une solution plus précise est de générer les tensions à comparer à la tension à mesurer par un CNA. Deux possibilités s'offrent alors : Soit tester toutes les valeurs possibles. La logique de commande est alors un compteur d'impulsions. Soit utiliser la dichotomie (division de l'espace de recherche de moitié). La logique de

commande est alors un registre à décalage : on essaye d'abord 1000000000, puis on essaie le bit suivant... Sur 16 bits il faut 16 tests par dichotomie au lieu de 65536.

La troisième solution est de charger un RC par la tension à mesurer en comptant le temps nécessaire pour arriver à un seuil donné. On choisit les valeurs de R et C de manière à obtenir une caractéristique linéaire. Le principal intérêt est que l'on ne mesure pas une valeur instantanée mais une quantité de courant (intégration). Il est donc inutile de filtrer les bruits qui sont superposés au signal (si ils sont faibles, haute fréquence et valeur moyenne nulle). Pour plus de précision, on utilise une double rampe : on charge le RC pendant un temps donné par la tension à mesurer, puis on le décharge par une tension connue, en mesurant le temps nécessaire. Par cette méthode, les imprécisions se soustraient au lieu de s'additionner.

9.3 échantillonnage

On appelle échantillonneur bloqueur un composant qui lit une valeur analogique à un instant donné, puis la mémorise (dans une capacité) jusqu'à la lecture. On échantillonne une valeur analogique à une fréquence au moins double de la plus petite fréquence à mesurer.

Pour transmettre un signal, il est plus efficace de le faire en numérique (peu ou pas de déformation de l'information, facilité de remise en forme du signal en cours de transmission longue distance) :

CAN −> multiplexeur −> liaison série −> démultiplexeur −> CNA

Mais on peut également utiliser une représentation stochastique :

On échantillonne, et on envoie un NOMBRE d'impulsions proportionnel à la valeur, de durée constante, réparties à peu près régulièrement (par exemple aléatoirement). On récupère alors ce signal en l'intégrant (c'est à dire enregistrer dans un RC la somme du courant arrivé en un temps donné). Cette méthode évite de synchroniser l'émetteur et le récepteur (bits de start et de stop, vitesse de transmission...)

10 ANNEXES : les transparents

T1. l'algèbre de BOOLE vérifié pour les circuits électriques et les ensembles

T2. Applications pneumatiques

T3. Application du combinatoire : l'additionneur

T4. Avantage du binaire réfléchi : les codeurs incrémentaux et absolus

T5. Représentation des portes

T6. Codeur 8 dans 3

T7. Bascules RS (à NOR ou NAND)

T8. Bascules RST / MS

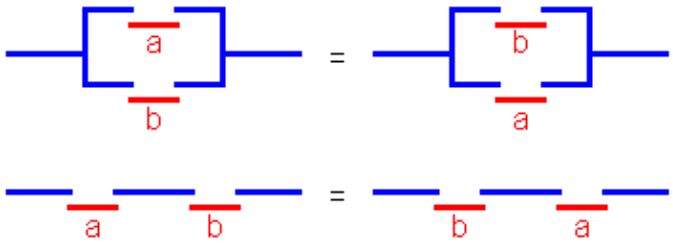
T9. Conversion numérique −> analogique

T10. Conversion analogique −> numérique

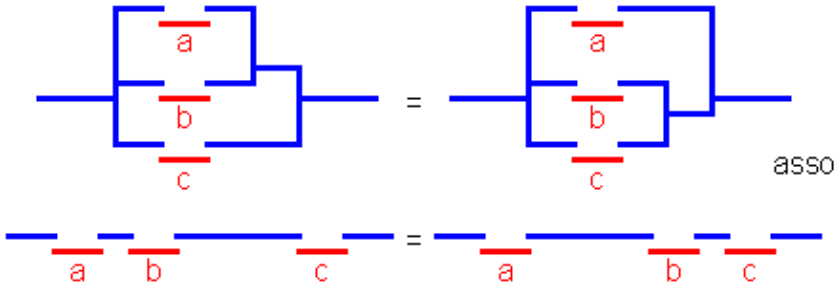
Depuis Janvier 98 mes transparents sont enfin disponibles (les liens ci−dessus n'étaient pas valides auparavant). Suite à de nombreux mails me demandant (gentiment) de les numériser, je l'ai enfin fait. Donc le lien qui suit, vers ma messagerie, n'a plus lieu d'être. Je le laisse néanmoins, au cas où vous désireriez me faire plaisir : − freeware var linktext = "envoyez moi un mail"; var nom = "Patrick.Trau"; var srv = "ipst−ulp.u−strasbg.fr"; document.write("" + linktext + "") //−−> pour me dire que vous avez lu ce document jusqu'ici.

11 Les axiomes de l'algèbre de Boole
Appliqués aux circuits électriques

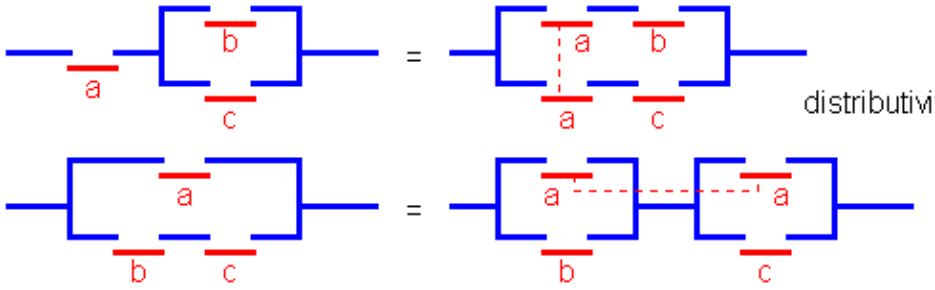
Hypothèse : 1=le courant peut passer (0 il ne peut pas) opérateur ET : série, OU : parallèle



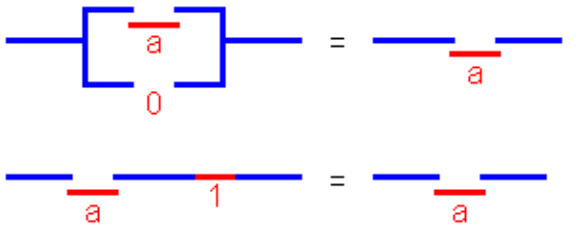
commutativité



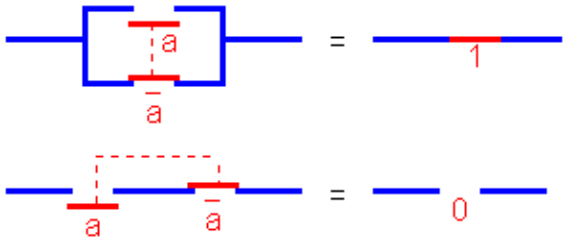
associativité



distributivité



éléments neutres



complémentation

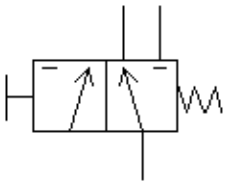
Remarques :

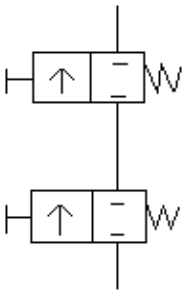
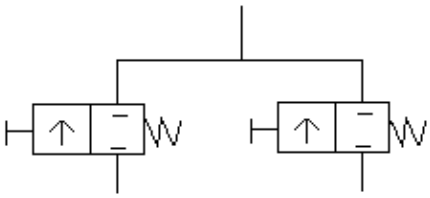
1) vous pouvez vérifier que les théorèmes s'appliquent bien

2) les équivalences ci−dessus ne sont vraies que pour les états stables. Dans les phases transitoires, il peut y avoir des comportements indésirables appelés **aléas technologiques**. Prenons l'exemple de la distributivité : les deux contacts a (interrupteur double circuit) ne peuvent pas changer d'état (contact ou non) exactement au même moment. C'est l'état obtenu entre ces deux instants que l'on appelle aléa technologique.

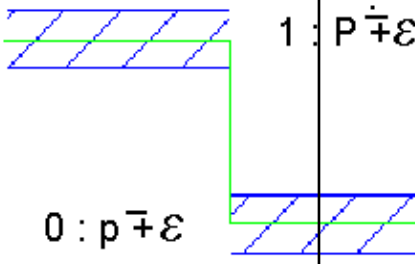
12 Transparent T2

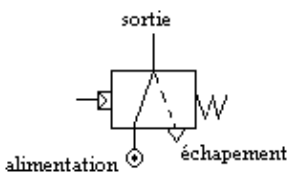
12.1 Pneumatique, convention 1 : le fluide peut passer ou non

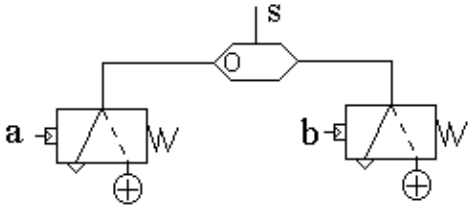
	Représentation d'un distributeur 3/2 (3 orifices, 2 positions), à commande mécanique (symbole à gauche), monostable (symbole à droite)

	
fonction ET (en série) (distributeur 2/2 monostable à commande mécanique)	fonction OU (en parallèle). Il n'y a pas de nécessité d'implanter un clapet anti−retour.

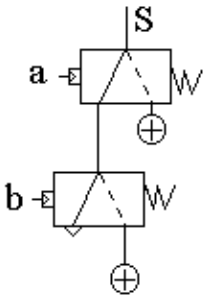
12.2 Pneumatique, convention 2 :

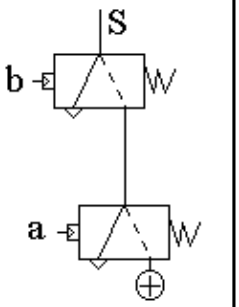
1=il y a la pression P prévue (avec un intervalle de tolérance), 0 = niveau p (par exemple pression atmosphérique) :	

	complémentation : la sortie est complémentaire à la commande (on utilise un distributeur monostable à commande pneumatique)



OU : on doit utiliser un clapet anti retour, sinon toute la pression s'échappe par le second distributeur. La seule solution est d'éviter les recouvrements sur le tableau de Karnaugh : $s = a + \bar{b}a$ tel que ci−contre



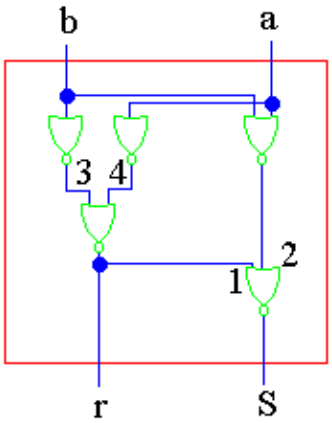


ET : en série, pas de problème

Patrick TRAU, ULP − IPST, 13/1/98

13 Transparent 3
Application combinatoire : l'additionneur

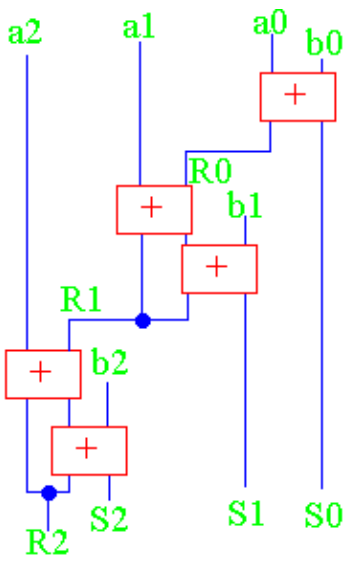
Trouvez l'équation des sorties r et s en fonction des entrées a et b



Rappel :

0d=0b
1d=1b
2d=10b
3d=11b
4d=100b
... etc...

c'est un additionneur 1 bit



Au maximum, a1+b1+R0 = 1+1+1

Donc on a au maximum une retenue de 1.

Donc R1 ne peut pas provenir des deux additionneurs en même temps.

Il n'empêche qu'en toute rigueur les deux noeuds représentés dans le schéma doivent être des portes OU

Ce composant est un additionneur 3bits :

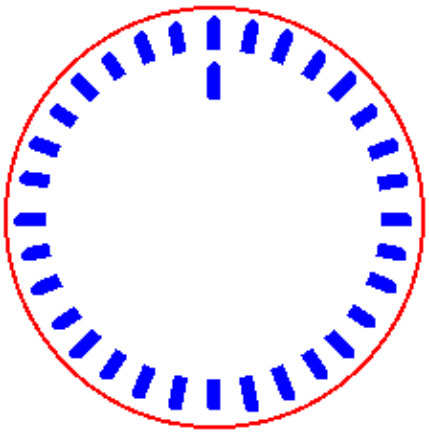
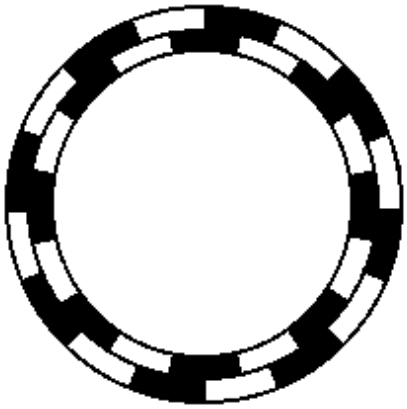
$$\begin{array}{r} a_2 a_1 a_0 \\ + \quad b_2 b_1 b_0 \\ \hline R_2 s_2 s_1 s_0 \end{array}$$

Patrick TRAU, ULP − IPST, 14/1/98

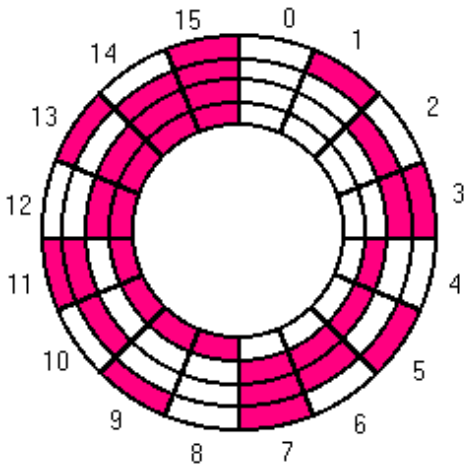
14 Capteurs de position angulaire
Intérêt du code GRAY (binaire réfléchi)

Transparent 4

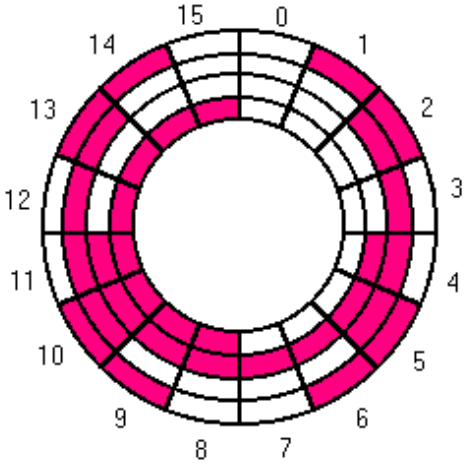
Capteur incrémental : nécessité d'un capteur supplémentaire pour le zéro, et d'une procédure d'initialisation ("homing") en général longue. Impossibilité de voir si l'on a "sauté" un pas (choc, vitesse excessive..)

	 En sens trigo : 00, 01, 11, 10, etc. Ca ne vous dit rien ?
Cas simple : impossible de voir le sens	déterminez le composant qui donne le sens en fonction de 4 entrées : la valeur actuelle et précédente des deux capteurs

Capteur absolu : à tout moment on connaît la position

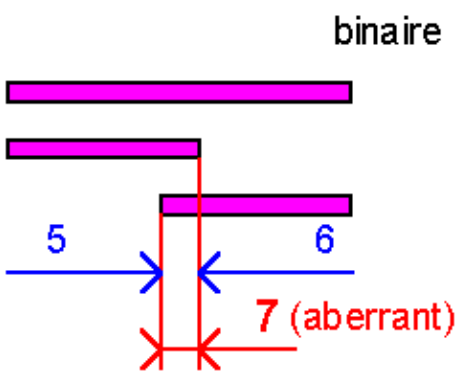
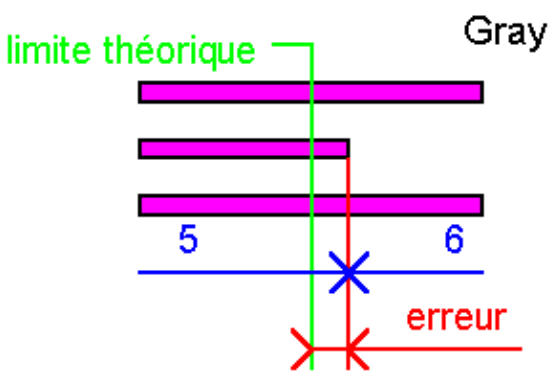


Binaire



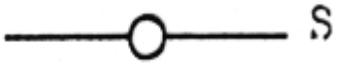
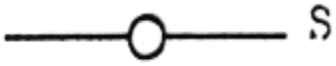
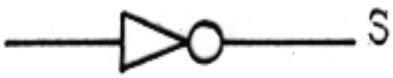
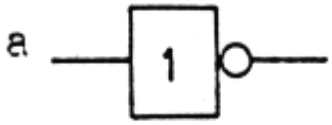
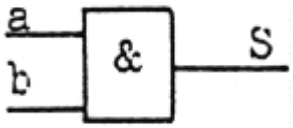
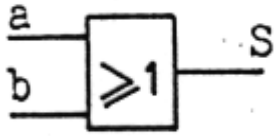
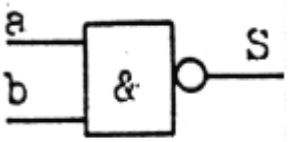
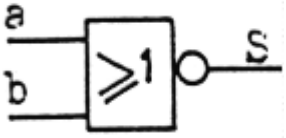
Binaire réfléchi (Gray)

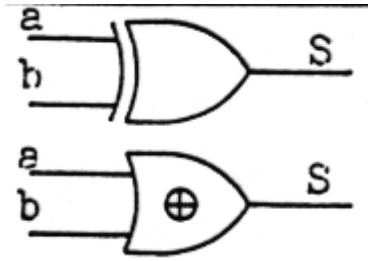
Le traçage des pistes ne peut pas se faire de façon parfaite : il y a un intervalle de tolérance. Analysons le passage de 5 à 6 :

	
passage par une valeur FAUSSE : c'est inadmissible	position JUSTE, mais avec un intervalle de tolérance : acceptable à condition de connaître la précision.

Patrick TRAU, ULP − IPST, 14/1/98

15 SYMBOLES DES OPERATEURS LOGIQUES

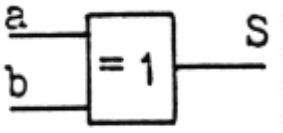
15.0.1 Symbole (Norme MILSTD 086B)	**15.0.2 Equations**	**15.0.3 Symbole** (notation française)
	Ensemble quelconque (la fonction est notée ou symbolisée à l'intérieur)	
	Inverseur	
	Ampli inverseur	
	Opérateur ET (AND) $S = a.b$	
	Opérateur OU (OR) $S = a + b$	
	Opérateur NON ET (NAND) $S = \overline{a.b} = \overline{a} + \overline{b}$	
	Opérateur NON OU (NOR) $S = \overline{a + b} = \overline{a}.\overline{b}$	



Opérateur OU Exclusif (XOR)

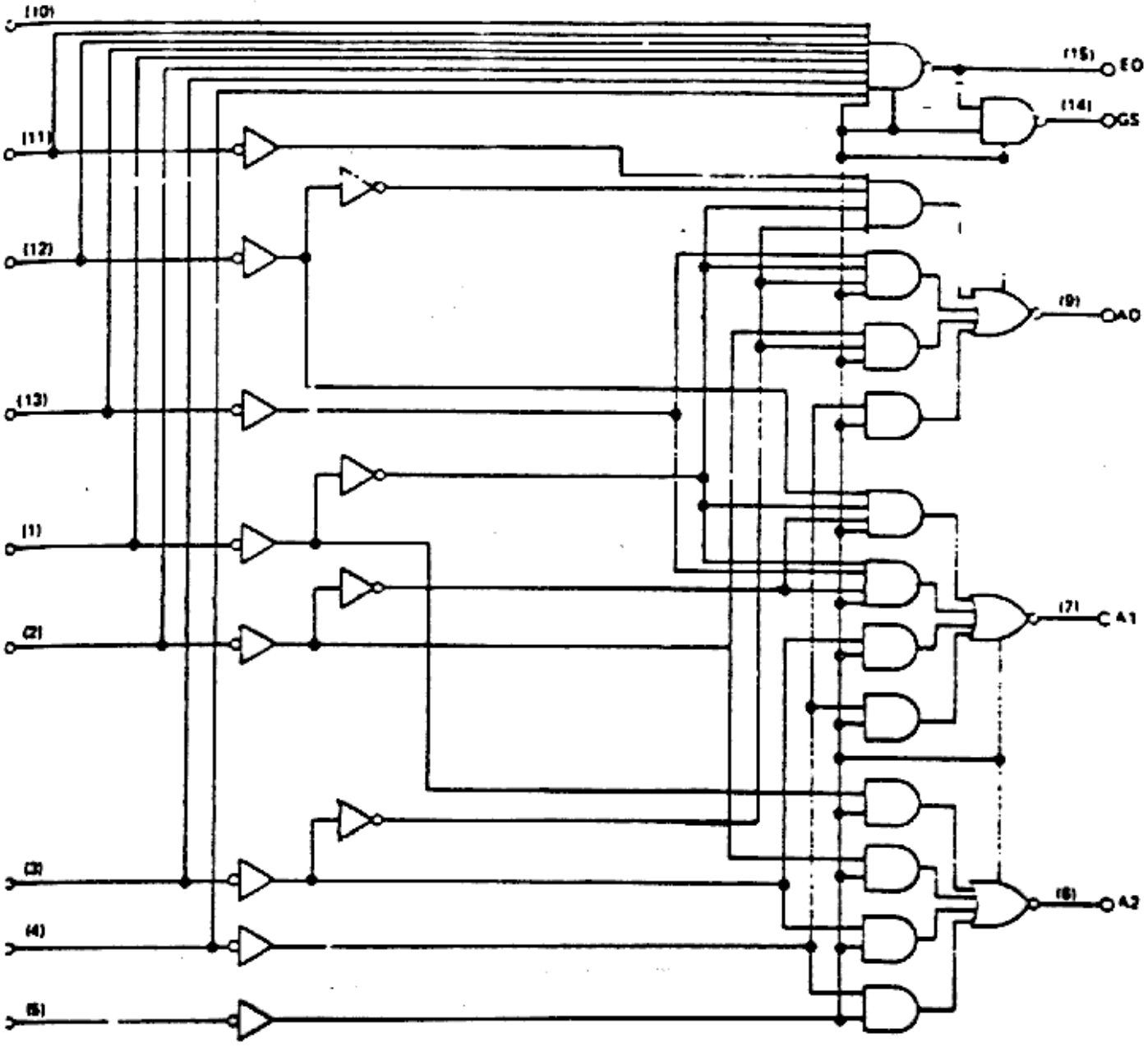
$$S = a\bar{b} + \bar{a}b$$

$$S = a \oplus b$$

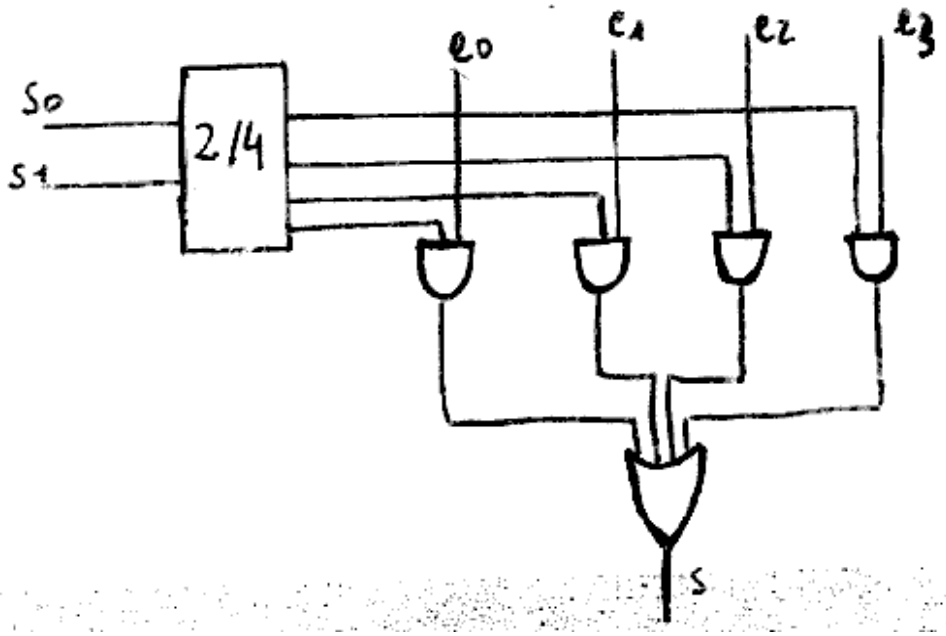


Patrick TRAU, ULP – IPST, 14/1/98

16 Encodeur de priorité 8 donne 3

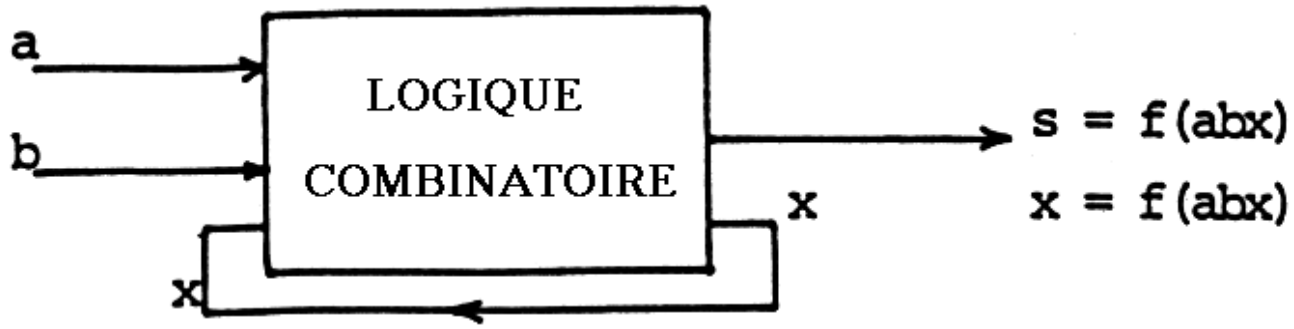


17 Multiplexeur 2 −>4

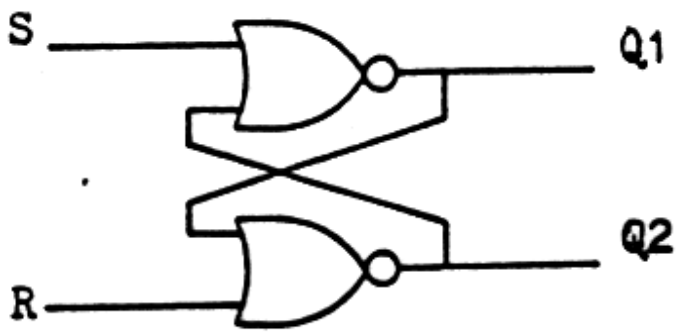


Patrick TRAU, ULP − IPST, 14/1/98

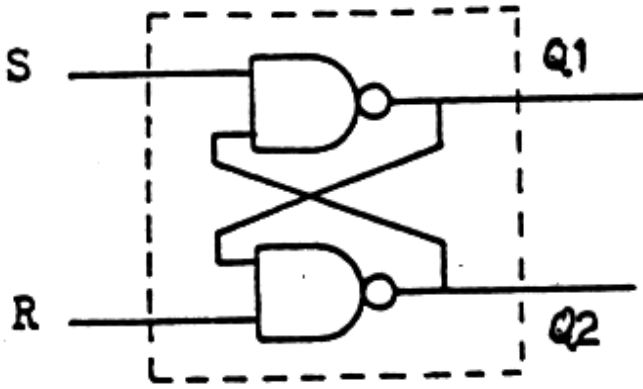
18 transparent



7

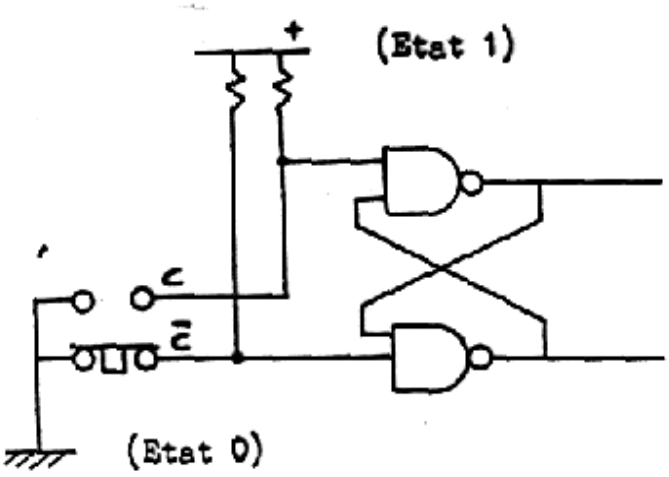


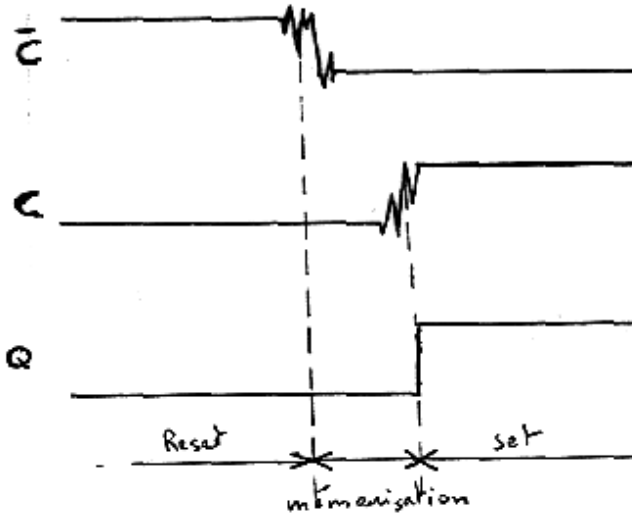
Bascule RS (à base de NOR)



Bascule RS (à base de NAND)

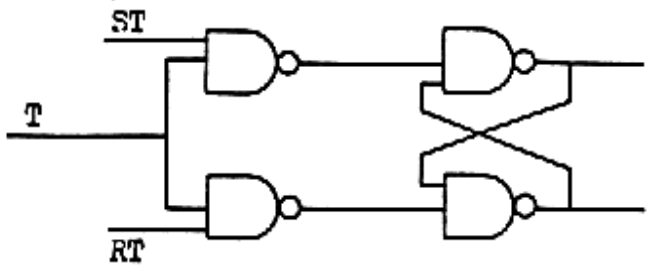
18.1 Circuit anti−rebond



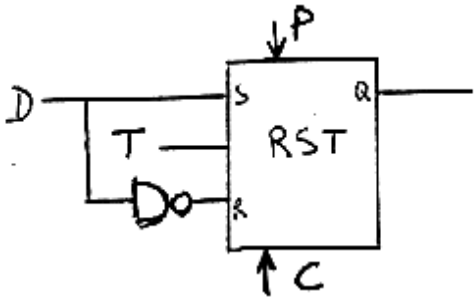


19 Transparent 8

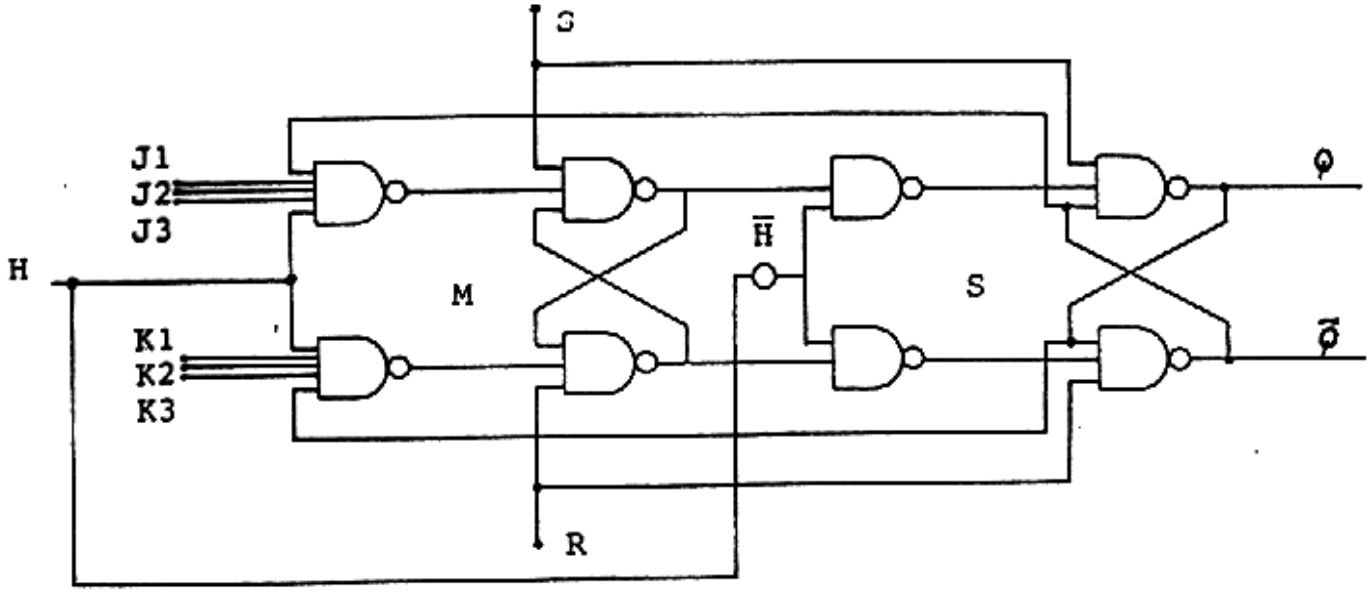
19.1 Bascule RST :



19.2 Bascule D :

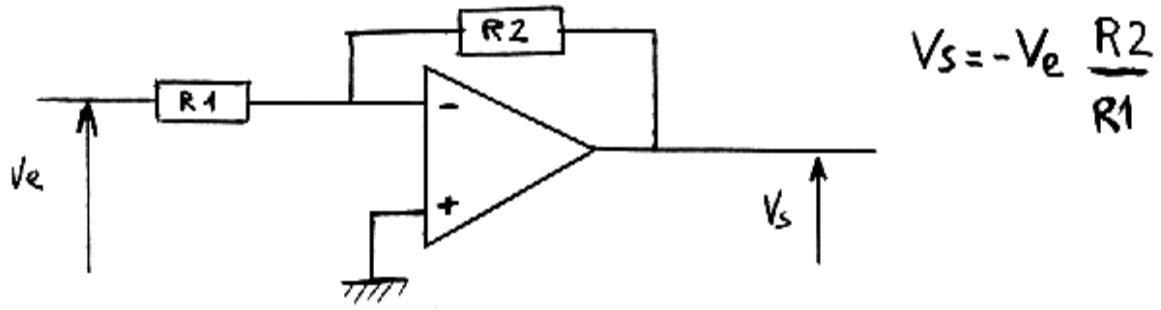


19.3 Bascule JK (Maître Esclave)

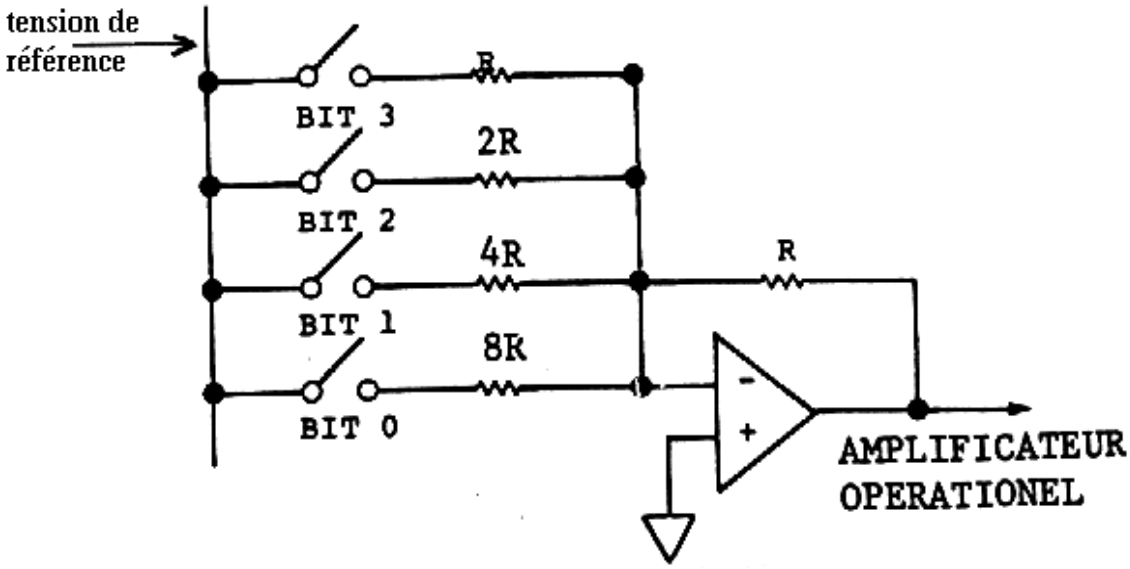


20 Transparent 9

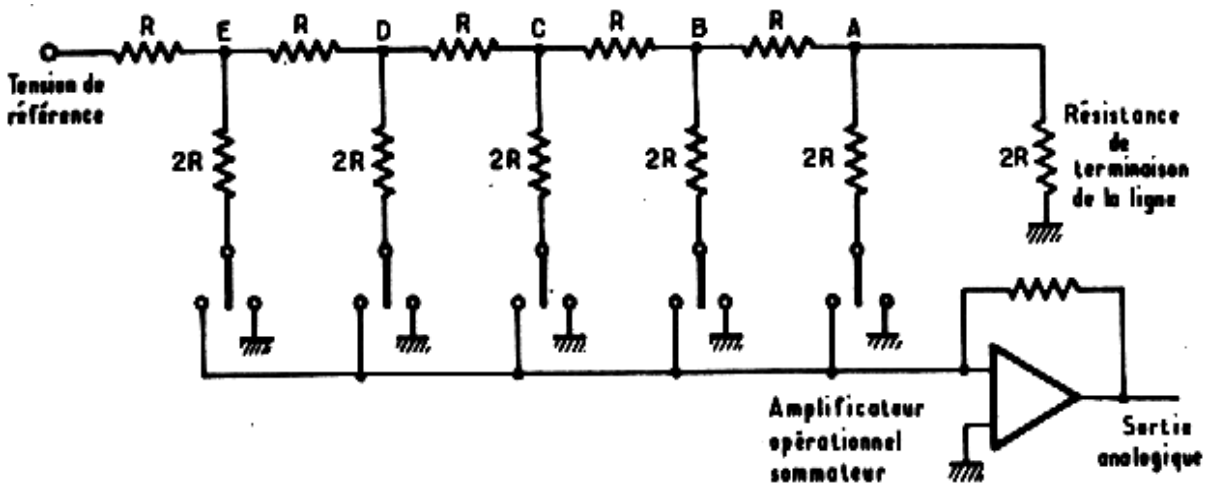
20.1 le comparateur



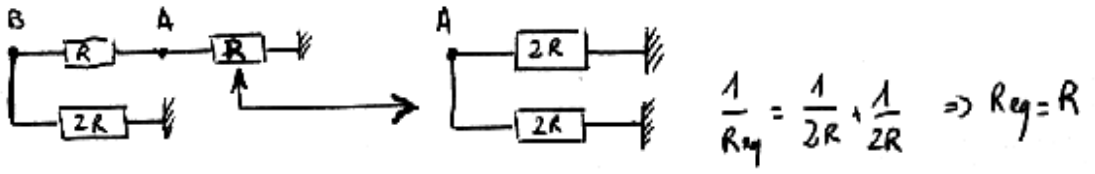
20.2 CNA 4 bits :



20.3 CNA de type R−2R (sur 5 bits) :

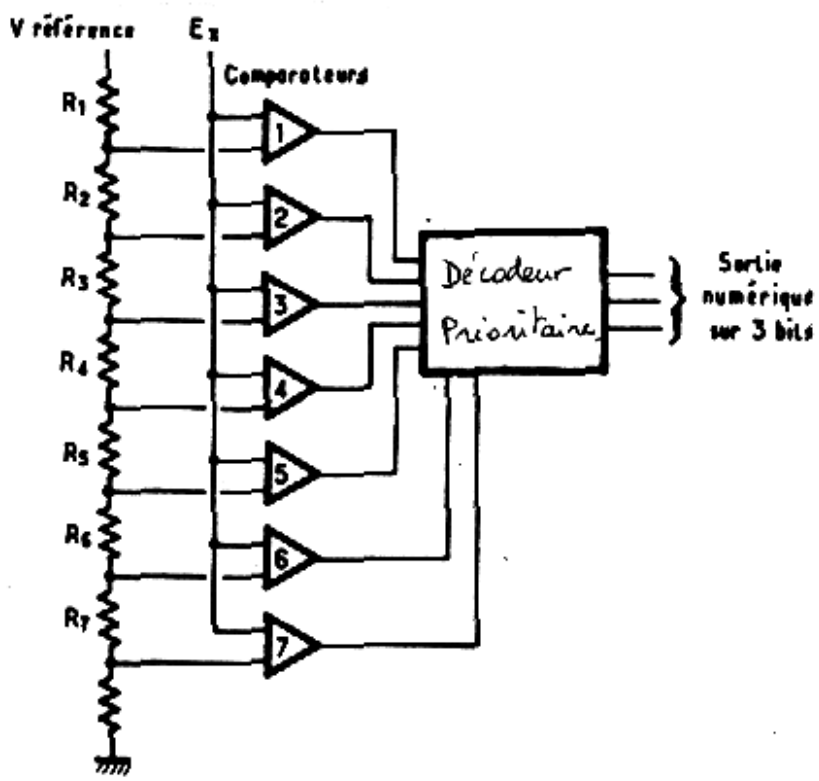


explication : à chaque noeud, l'intensité est divisée par 2 :

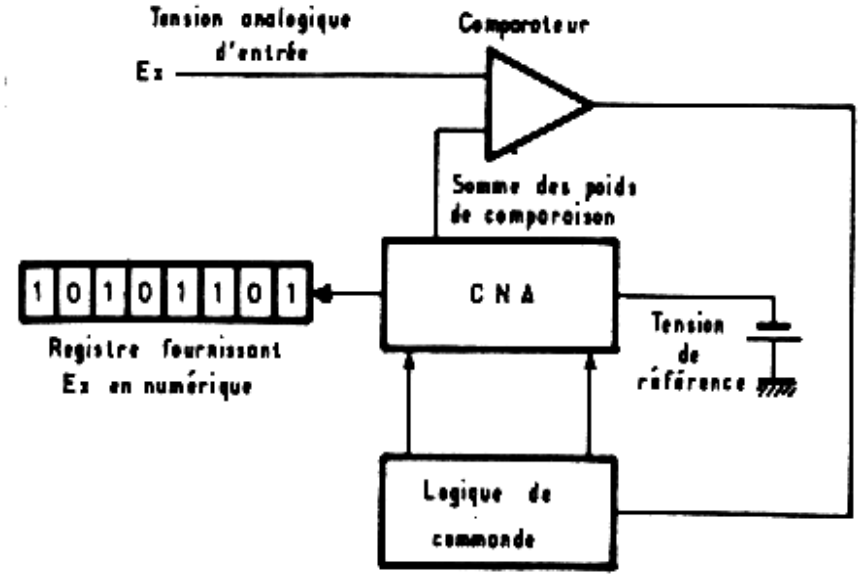


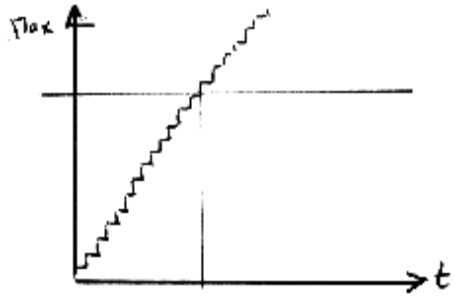
21 Transparent 10

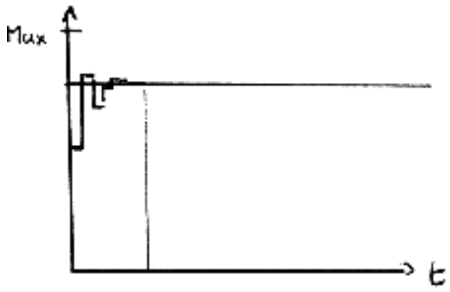
21.1 CAN (Convertisseur Analogique −> Numérique) direct



21.2 CAN à l'aide d'un CNA :

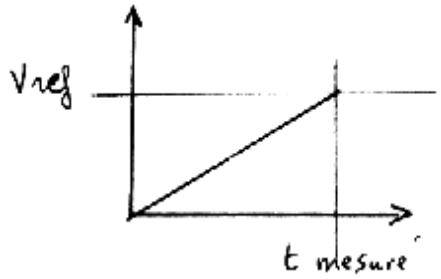


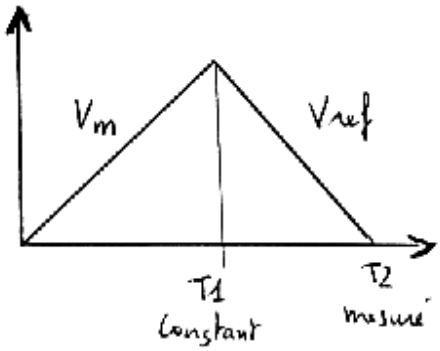




logique de commande = compteur dichotomie (utilisation d'un registre à décalage)

à l'aide d'un compteur et d'une horloge, on compte le temps de charge d'un RC :



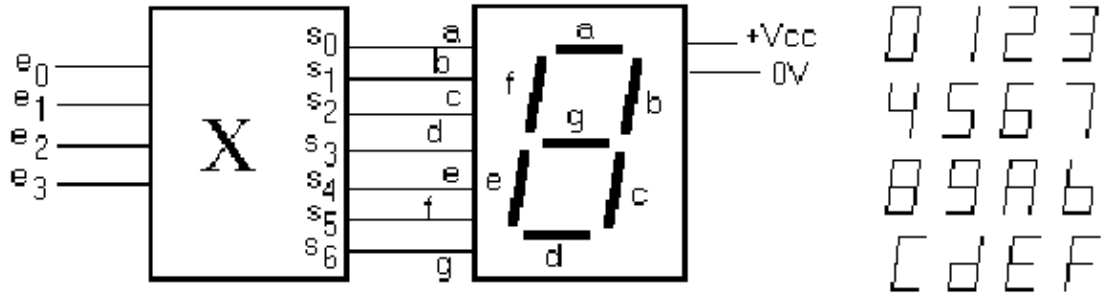


$$\frac{V_m}{V_{ref}} = \frac{T_2}{T_1}$$

Simple rampe double rampe

22 afficheur 7 segments

22.1 Enoncé du problème



Trouver le schéma du composant X. Ses 4 entrées correspondent à la représentation binaire d'un chiffre entre 0 et 15. Il faut fournir en sortie les 7 signaux nécessaires à l'affichage du chiffre hexadécimal correspondant. On suppose qu'il faut un 0 pour allumer un segment, et un 1 pour l'éteindre.

22.2 table de vérité

regroupons dans un table l'état désiré pour les sorties dans chaque cas. L'ordre n'a pas d'importance

décimal		hexa		binaire					a	b	c	d	e	f	g
0		0		0	0	0	0		0	0	0	0	0	0	1
1		1		0	0	0	1		1	0	0	1	1	1	1
2		2		0	0	1	0		0	0	1	0	0	1	0
3		3		0	0	1	1		0	0	0	0	1	1	0
4		4		0	1	0	0		1	0	0	1	1	0	0
5		5		0	1	0	1		0	1	0	0	1	0	0
6		6		0	1	1	0		0	1	0	0	0	0	0
7		7		0	1	1	1		0	0	0	1	1	1	1
8		8		1	0	0	0		0	0	0	0	0	0	0
9		9		1	0	0	1		0	0	0	0	1	0	0
10		A		1	0	1	0		0	0	0	1	0	0	0
11		B		1	0	1	1		1	1	0	0	0	0	0
12		C		1	1	0	0		0	1	1	0	0	0	1
13		D		1	1	0	1		1	0	0	0	0	1	0
14		E		1	1	1	0		0	1	1	0	0	0	0
15		F		1	1	1	1		0	1	1	1	0	0	0

22.3 recherche des équations

On peut maintenant analyser chaque sortie indépendamment, pour déterminer les équations. Nous allons utiliser des tableaux de Karnaugh

e1e0

e3e2

a	00	01	11	10
00	0	1	0	0
01	1	0	0	0
11	0	1	0	0
10	0	0	1	0

$$a = \overline{e_1}e_0\ (\overline{e_3}\overline{e_2} + e_3e_2) + \overline{e_3}e_2\overline{e_1}\overline{e_0} + e_3\overline{e_2}e_1e_0$$

e1e0

e3e2

b	00	01	11	10
00	0	0	0	0
01	0	1	0	1
11	1	0	1	1
10	0	0	1	0

$$b = e_2\overline{e_0}\ (e_1+e_3) + e_3e_1e_0 + \overline{e_3}e_2\overline{e_1}e_0$$

e1e0

e3e2

c	00	01	11	10
00	0	0	0	1
01	0	0	0	0
11	1	0	1	1
10	0	0	0	0

$$c = e_3e_2\ (e_1+\overline{e_0}) + \overline{e_3}\overline{e_2}e_1\overline{e_0}$$

e1e0

e3e2

d	00	01	11	10
00	0	1	0	0
01	1	0	1	0
11	0	0	1	0
10	0	0	0	1

$$d = e_2e_1e_0 + \overline{e_3}\overline{e_2}\overline{e_1}e_0 + \overline{e_3}e_2\overline{e_1}\overline{e_0} + e_3\overline{e_2}e_1\overline{e_0}$$

e1e0

e3e2

e	00	01	11	10
00	0	1	1	0
01	1	1	1	0
11	0	0	0	0
10	0	1	0	0

$$e = \overline{e_3}\,(e_0 + e_2\overline{e_1}) + \overline{e_2}\,\overline{e_1}e_0$$

e1e0

e3e2

f	00	01	11	10
00	0	1	1	1
01	0	0	1	0
11	0	1	0	0
10	0	0	0	0

$$f = \overline{e_3}\,\overline{e_2}(e_1+e_0) + \overline{e_3}e_1e_0 + e_3e_2\overline{e_1}e_0$$

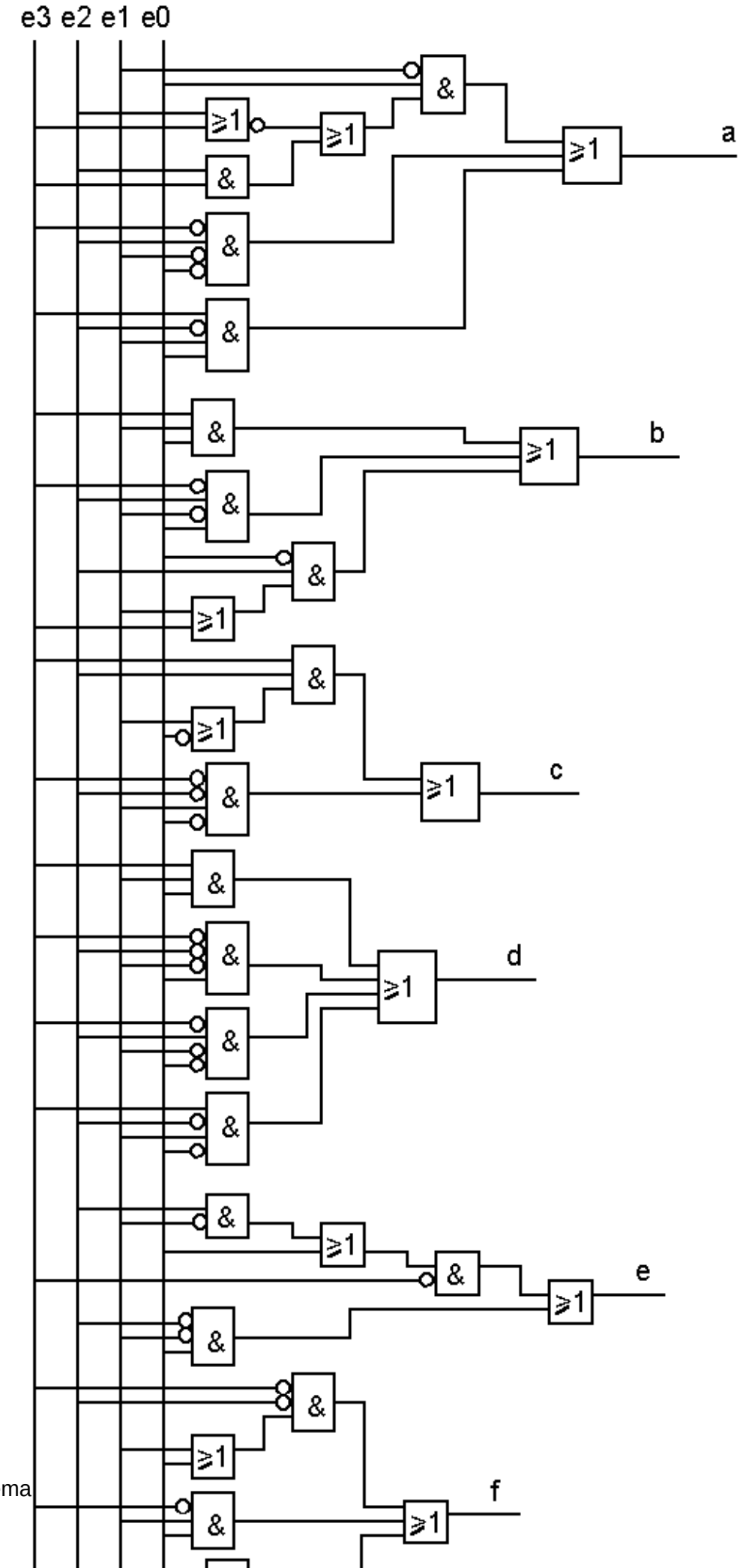
e1e0

e3e2

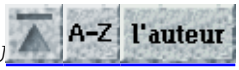
g	00	01	11	10
00	1	1	0	0
01	0	0	1	0
11	1	0	0	0
10	0	0	0	0

$$g = \overline{e_3}\,(\overline{e_2}\,\overline{e_1} + e_2e_1e_0) + e_3e_2\overline{e_1}\,\overline{e_0}$$

22.4 schéma


e3 e2 e1 e0
a
b
c
d
e
f

Dernière mise à jour le 7/11/2000

Par Patrick TRAU 

23 Bases d'automatisme − Sommaire