

Cours 1 - La numération

I - Définitions

I-1) Expression générale

La base b d'un système de numération représente le nombre d'unités d'un certain rang, nécessaires pour former une unité de rang immédiatement supérieur.

L'ensemble $B_b = [0, 1, 2, \dots, b-1]$, soit b caractères (chiffres en base 10) quantifie le nombre d'unités d'un rang quelconque.

Tout nombre en base b s'écrit $N_b = a_n \dots a_1 a_0 a_{-1} \dots a_m$ et s'exprime en base 10 à l'aide de la base b :

$$(E-1) \quad N_{10} = a_n \cdot b^n + a_{n-1} \cdot b^{n-1} + \dots + a_0 \cdot b^0 + a_{-1} \cdot b^{-1} + \dots + a_m \cdot b^m$$

On peut aussi l'exprimer en base b à l'aide de la base 10 :

$$(E-2) \quad N_b = a_n \cdot (10)_b^n + a_{n-1} \cdot (10)_b^{n-1} + \dots + a_0 \cdot (10)_b^0 + a_{-1} \cdot (10)_b^{-1} + \dots + a_m \cdot (10)_b^m$$

où a_i est un chiffre de base b positionné au rang i . n et m sont les rangs extrêmes. $n > 0$ pour la partie entière et $m < 0$ pour la partie fractionnaire.

b est un nombre écrit en base 10 et b_i exprime le poids du rang i .

Remarque : $(10)_b = (b)_{10}$ et d'autre part a_i est compris entre 0 inclus et $(b)_{10}$ exclu.

I-2) Rappels : la numération décimale.

La base $b=10$, donc $B_b = [0, 1, \dots, 9]$. Ici N_{10} et N_b sont confondus. Soient quelques exemples :

$$N_{10} = (5807)_{10} = (5 \cdot 10^3 + 8 \cdot 10^2 + 0 \cdot 10^1 + 7 \cdot 10^0)_{10} = (5000 + 800 + 7)_{10}$$

$$N_{10} = (4,75)_{10} = (4 \cdot 10^0 + 7 \cdot 10^{-1} + 5 \cdot 10^{-2})_{10} = (4 + 0,7 + 0,05)_{10}$$

II - Système de numération binaire.

II-1) Expression

Ce système créé par Leibnitz (17e s) utilise la base 2 donc $B_b = [0, 1]$. Chaque nombre se présente ainsi :

$$N_2 = (100,11)_2 = (1 \cdot 10_2^2 + 0 \cdot 10_2^1 + 0 \cdot 10_2^0 + 1 \cdot 10_2^{-1} + 1 \cdot 10_2^{-2})_2$$

A partir de l'équivalence $b = (2)_{10} = (10)_2$, il vient :

$$N_2 = (100,11)_2 = (1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2})_2$$

Numération binaire :

$$N_{10} = a_n \cdot 2^n + a_{n-1} \cdot 2^{n-1} + \dots + a_0 \cdot 2^0 + a_{-1} \cdot 2^{-1} + \dots + a_m \cdot 2^m$$

II-2) Conversion Binaire-Décimal

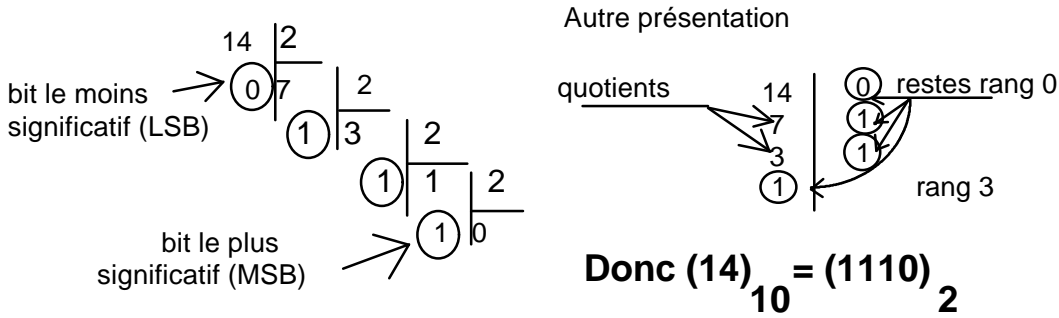
Méthode : à partir de l'équivalence donnée ci-dessus, la conversion de N_2 s'obtient en décomposant le nombre en une somme de termes selon l'expression (E-1).

Exemple : $N_2 = (100,11)_2 \rightarrow N_{10} = 1 \cdot 2^2 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2} = (4 + 0,5 + 0,25)_{10} = (4,75)_{10}$

II-3) Conversion Décimal-Binaire.

Méthode : On effectue une suite de divisions successives du nombre N_{10} par la base b , les restes obtenus constituant à chaque rang le caractère a_i du nombre N_2 .

Exemple : $N_{10} = (14)_{10}$.

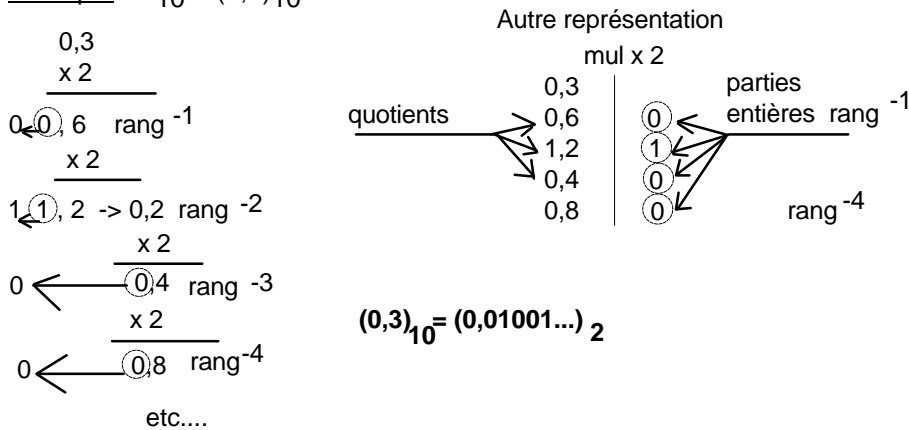


Conversion par décomposition en puissance de 2 (cas des petits nombres)
 $(14)_{10} = (0.16) + (1.8) + (1.4) + (1.2) + (0.1) = (01110)_2$.

II-4) Cas des nombres fractionnaires.

Méthode : On effectue une suite de multiplications successives de N_{10} par la base b, de la partie fractionnaire seulement, la partie entière des résultats obtenus constituant à chaque rang le caractère ai du nombre N_2 .

Exemple : $N_{10} = (0,3)_{10}$.



Remarque : cette conversion ne s'arrête pas car dans la représentation binaire, on ne peut exprimer exactement tous les nombres fractionnaires de N_{10} . Il faut donc se limiter à un format en respectant la précision décimale souhaitée (ex 10^{-5}).

II-5) Précision fractionnaire.

Pour un nombre donné en base 10, avec une précision de 10^{-n} près, on effectue une conversion binaire telle que l'unité du rang du dernier chiffre obtenu soit égale ou inférieure à celle du dernier rang donné en base 10. Donc pour la base 2 : $1.2^{-m} \leq 1.10^{-n} \Leftrightarrow \log_{10} 2^{-m} \leq \log_{10} 10^{-n}$

Soit $m \geq n/\log_{10} 2 = n. 3,32$

Preons un exemple. Soit $(0,3)_{10}$ à 10^{-4} près.

$m \geq 4.3,32$ soit 13,28, c'est à dire 14 chiffres après la virgule. En binaire :

$(0,3)_{10} = (0,01001100110001)_2$.

II-6) Utilisation pour la longueur de conversion décimale-binaire.

Ce résultat peut être utilisé dans la détermination du format du nombre binaire équivalent (occupation et longueur des registres binaires). En effet, si N_{10} a une longueur de n chiffres et si N_2 a une longueur de m chiffres, pour des nombres entiers on a : $m \geq 3,32 n$

Exemples :

$(32)_{10} = (100000)_2$ $n=2$ $m=6$ $m/n=3$

$(500)_{10} = (111110100)_2$ $n=3$ $m=9$ $m/n=3$

$(64)_{10} = (1000000)_2$ $n=2$ $m=7$ $m/n=3,5$

Dans ce dernier cas, le registre binaire sera de longueur directement supérieure à $(3.32 n)$

III - Autres systèmes de numérations.

III-1) Système de numération octale.

La base est $b=8$ et l'ensemble $B_8 = [0,1,2,\dots,7]$. Il y a équivalence entre :
 $b=8_{10}=10_8=(1000)_2$ et $8^n=2^{3n}$

Expression : $N_8=(13257)_8=(1.10_8^4+3.10_8^3+2.10_8^2+5.10_8^1+7.10_8^0)_8$ ou encore :
 $N_8=(265,42)_8=(2.10_8^2+6.10_8^1+5.10_8^0+4.10_8^{-1}+2.10_8^{-2})$

Conversion : à partir des relations d'équivalence ci-dessus donne $(5807)_{10}$ pour le premier et $(181,53125)_{10}$ pour le deuxième.

Conversion décimale octale : La méthode est analogue à celle employée pour la conversion décimale binaire. On pourra aussi utiliser des tables de conversion.

Exemple :

Nombre entier	Nombre fractionnaire																				
div 8	x8																				
<table border="0" style="margin: auto;"> <tr><td style="padding-right: 10px;">(5807)</td><td style="border-left: 1px solid black; padding-left: 10px; text-align: center;">⑦</td></tr> <tr><td style="padding-right: 10px;">725</td><td style="border-left: 1px solid black; padding-left: 10px; text-align: center;">⑤</td></tr> <tr><td style="padding-right: 10px;">90</td><td style="border-left: 1px solid black; padding-left: 10px; text-align: center;">②</td></tr> <tr><td style="padding-right: 10px;">11</td><td style="border-left: 1px solid black; padding-left: 10px; text-align: center;">③</td></tr> <tr><td style="padding-right: 10px;">①</td><td style="border-left: 1px solid black; padding-left: 10px;"></td></tr> </table>	(5807)	⑦	725	⑤	90	②	11	③	①		<table border="0" style="margin: auto;"> <tr><td style="padding-right: 10px;">0,83</td><td style="border-left: 1px solid black; padding-left: 10px; text-align: center;">⑦</td></tr> <tr><td style="padding-right: 10px;">6,64</td><td style="border-left: 1px solid black; padding-left: 10px; text-align: center;">⑥</td></tr> <tr><td style="padding-right: 10px;">5,12</td><td style="border-left: 1px solid black; padding-left: 10px; text-align: center;">⑤</td></tr> <tr><td style="padding-right: 10px;">0,96</td><td style="border-left: 1px solid black; padding-left: 10px; text-align: center;">⑦</td></tr> <tr><td style="padding-right: 10px;">7,68</td><td style="border-left: 1px solid black; padding-left: 10px; text-align: center;">⑦</td></tr> </table>	0,83	⑦	6,64	⑥	5,12	⑤	0,96	⑦	7,68	⑦
(5807)	⑦																				
725	⑤																				
90	②																				
11	③																				
①																					
0,83	⑦																				
6,64	⑥																				
5,12	⑤																				
0,96	⑦																				
7,68	⑦																				
$(5807)_{10} = (13257)_8$	$(0,83)_{10} = (0,6507)_8$																				

Remarque : 1) cette base qui est une puissance de 2 n'utilise souvent pas plus de chiffres qu'en décimal.
 2) La précision fractionnaire est ici $m \geq n/\log_{10}8 = 1,1.n$

III-2) Système de numération hexadécimale.

La base est $b=16$ et l'ensemble $B_{16}=[0,1,2,\dots,9,A,B,C,D,E,F]$. Chaque lettre a une équivalence décimale (voir tableau).

Expression : Relations d'équivalence : $b=16_{10}=10_{16}=20_8=(10000)_2$ et $16^n=2^{4n}$
 $N_{16}=(16AF)_H = (1.10_{16}^3 + 6.10_{16}^2 + A.10_{16}^1 + F.10_{16}^0)$
 $N_{16} = (B5,23)_H = (B.10_{16}^1 + 5.10_{16}^0 + 2.10_{16}^{-1} + 3.10_{16}^{-2})$.

Conversion hexadécimal-décimal :

Une première méthode consiste à utiliser l'expression (E-1). On trouve ainsi $(5807)_{10}$ et $(181,137)_{10}$ pour les nombres proposés ci-dessus.

Numération hexadécimale :

$$N_2 = a_n \cdot 16^n + a_{n-1} \cdot 16^{n-1} + \dots + a_0 \cdot 16^0 + a_{-1} \cdot 16^{-1} + \dots + a_m \cdot 16^m$$

Une deuxième méthode consiste à utiliser un tableau de conversion permettant une rapidité de calcul ainsi qu'une simplification de celui-ci.

Exemple : $(A37F)_H = (A000 + 0300 + 0070 + 000F)_H = (40960 + 768 + 112 + 15)_{10} = (41855)_{10}$.

Nombres entiers hexadécimaux	Equivalents en base 10	nombres fractionnaires hexadécimaux	Equivalents en base 10
1	1	0,1	0,0625
2	2	0,2	0,125
3	3	0,3	0,1875
4	4	0,4	0,25
5	5	0,5	0,3125
6	6	0,6	0,375
7	7	0,7	0,4375
8	8	0,8	0,5
9	9	0,9	0,5625
A	10	0,A	0,625
B	11	0,B	0,6875
C	12	0,C	0,75
D	13	0,D	0,8125
E	14	0,E	0,875
F	15	0,F	0,9375
10	16	0,FFF FFF....	0,999 999
11	17		
12	18		
13	19		
14	20		
1E	30		
20	32		
32	50		
40	64		
64	100		
FF	255		
100	256		
12C	300		
1F4	500		
200	512		
3E8	1000		
FFF	4095		
1000	4096		

Conversion décimale hexadécimale : la méthode est toujours la même, mais il faut convertir les restes des divisions ou les parties entières des produits en hexadécimal lorsqu'ils sont compris entre 10 et 15.

Exemple :

<p>Nombre entier</p> <p>/16 :</p> $\begin{array}{r l} 5807 & \textcircled{F} \\ 362 & \textcircled{A} \\ 22 & \textcircled{6} \\ \textcircled{1} & \end{array}$ <p style="text-align: center;">$N_{16} = (16AF)_{16}$</p>	<p>Nombre fractionnaire</p> <p>x 16</p> $\begin{array}{r l} \textcircled{0},66 & \textcircled{0} \\ \textcircled{10},56 & \textcircled{A} \\ \textcircled{8},96 & \textcircled{8} \\ \textcircled{15},36 & \textcircled{F} \\ \textcircled{5},76 & \textcircled{5} \end{array}$ <p style="text-align: center;">$N_{16} = (0,A8F5)_{16}$</p>
---	---

Une autre méthode consiste à utiliser le tableau de conversion.

$$N_{10} = (5807)_{10} = (5000 + 800 + 7)_{10}$$

$$N_{16} = (1388 + 0320 + 0007)_{16} = (16AF)_{16}$$

l'addition est faite ici en hexadécimal.

On peut aussi passer par le binaire.

Remarques : 1) La base étant aussi une puissance de 2 mais qui ne nécessite cependant autant ou moins de caractères que la base dix pour la même quantité d'unités.

2) La précision fractionnaire est $m \geq n/\log_{10}16 = 0,83.n$.
 Dans l'exemple $(0,66)_{10}$ est à 10^{-2} près $m \geq 1,66$ donc $N_{16} = (0,A9)_{16}$

III-3) Conversions entre systèmes de numération (2, 8 ou 16).

$N_8 \rightarrow N_{16}$ On remplace chaque caractère de rang i de N_8 par son équivalent binaire.
Exemple : $(23,76)_8 = (010\ 011,111\ 110)_2$.

$N_{16} \rightarrow N_2$ même méthode.
Exemple : $(A8)_{16} = (1010\ 1000)_2$.

$N_2 \rightarrow N_8$ On remplace chaque groupe de 3 bits par son équivalent octal.
Exemple : $(101,110\ 010\ 1)_2 = (5,624)_8$.

$N_2 \rightarrow N_{16}$ même méthode mais par groupe de 4 bits de N_2 .
Exemple : $(0111\ 1010,1001\ 11)_2 = (7A,9C)_H$.

Applications : Ces conversions entre systèmes servent à :

- écrire sous forme condensée des mots de 8, 16 et 32 bits dans les programmes,
- convertir rapidement les nombres décimaux en binaire (passage par N_{16} ou N_8),
- effectuer des opérations.

HEXADECIMAL ----> DECIMAL

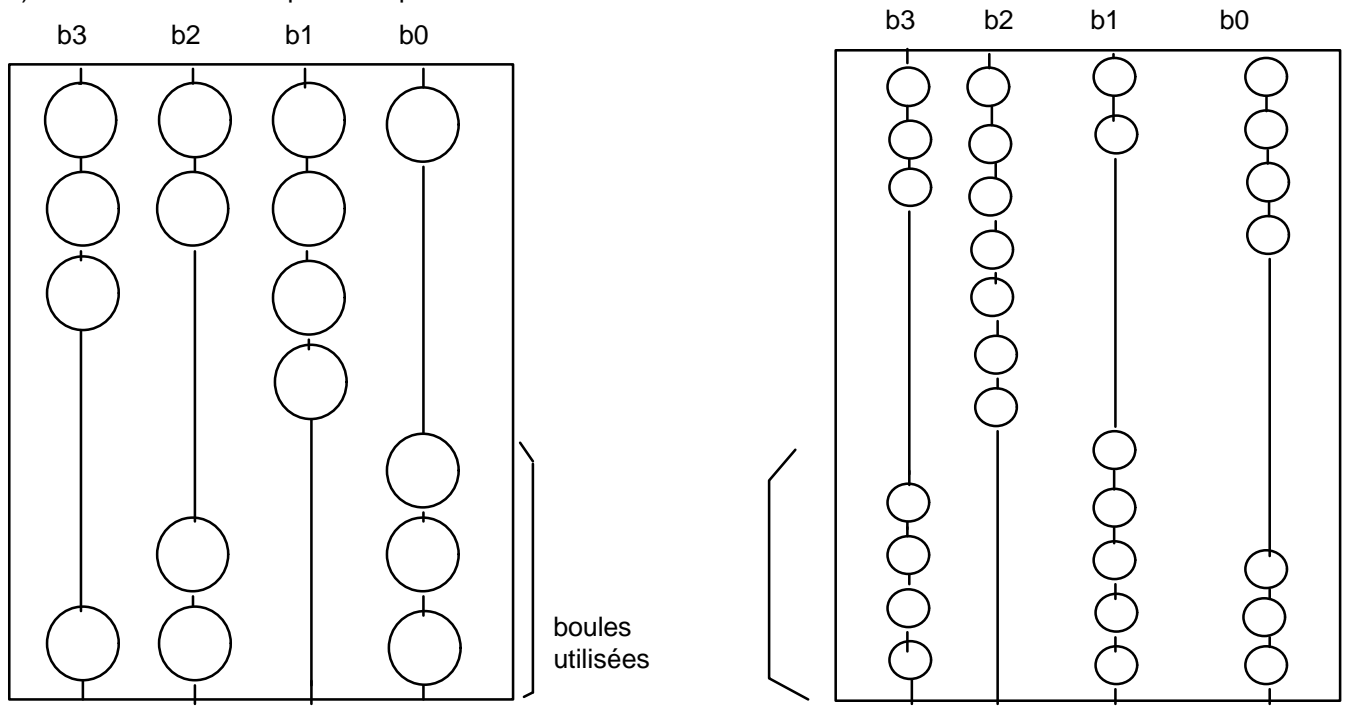
1000	4096	0100	256	0010	16	0001	1
2000	8192	0200	512	0020	32	0002	2
3000	12288	0300	768	0030	48	0003	3
4000	16384	0400	1024	0040	64	0004	4
5000	20480	0500	1280	0050	80	0005	5
6000	24576	0600	1536	0060	96	0006	6
7000	28672	0700	1792	0070	112	0007	7
8000	32768	0800	2048	0080	128	0008	8
9000	36864	0900	2304	0090	144	0009	9
A000	40960	0A00	2560	00A0	160	000A	10
B000	45056	0B00	2816	00B0	176	000B	11
C000	49152	0C00	3072	00C0	192	000C	12
D000	53248	0D00	3328	00D0	208	000D	13
E000	57344	0E00	3584	00E0	224	000E	14
F000	61440	0F00	3840	00F0	240	000F	15

DECIMAL ---> HEXADECIMAL

10000	2710	1000	03E8	100	0064	10	000A
20000	4E20	2000	07D0	200	00C8	20	0014
30000	7530	3000	0BB8	300	012C	30	001E
40000	9C40	4000	0FA0	400	0190	40	0028
50000	C350	5000	1388	500	01F4	50	0032
60000	EA60	6000	1770	600	0258	60	003C
		7000	1B58	700	02BC	70	0046
		8000	1F40	800	0320	80	0050
		9000	2328	900	0384	90	005A

IV - Exercices.

1) Trouver le nombre représenté par ces bouliers. Les écrire dans leur base et en base 10.



2) On dénombre en base 10 des éléments d'un ensemble. Ces nombres écrits en base b sont notés N_b . Dans quelle base sont-ils écrits ?

Soit 19 éléments $\rightarrow N_b = (201)_b \rightarrow$ base = ?

Soit 170 éléments $\rightarrow N_b = (442)_b \rightarrow$ base = ?

3) On compte en base b un ensemble de chaussures. Le nombre de paires est : $N_b = (352)_b$. Le nombre de chaussures est : $N_b = (724)_b$.

Découvrir dans quelle base s'est fait le dénombrement et quel est le nombre de chaussures exprimé en base 10.

4) Etude de conversions :

* Exprimer N_2 , N_{10} , N_8 pour $N_{16} = (ABCD)_{16}$; $(10)_{16}$; $(A07F)_{16}$; $(F6,B8)_{16}$; $(0,B2F)_{16}$.

* Exprimer N_2 , N_{16} , N_8 pour $N_{10} = 105$; 29 ; 35 ; 14625 ; 123,312.

* Exprimer N_{10} pour $N_8 = (476)_8$; $(350,32)_8$; $(0,327)_8$.

Cours 2 - Représentation des nombres.

Un nombre est représenté en format fixe par l chiffres dans sa base. Il s'écrit donc :

$$N_b = a_n a_{n-1} \dots a_1 a_0 \text{ avec } l = n+1.$$

La quantité de nombres de l chiffres qu'il est possible de représenter s'appelle : la capacité de représentation. C'est :

$$C = N_{10\max} + \text{représentation du zéro soit } C = N_{\max} + 1 = b^l \text{ et en exprimant } l \text{ (le format) :}$$

$$l = \log_b C = \ln(C) / \ln(b) \quad (\text{sera arrondie à la valeur supérieure})$$

Soient quelques exemples :

- La longueur des mots binaires étant $l=11$, quelle est la capacité de représentation C ?

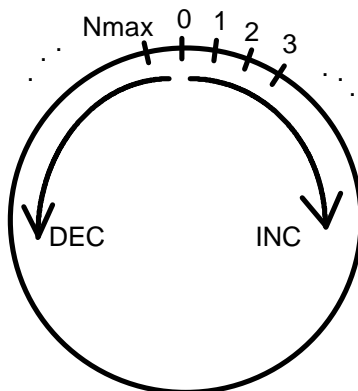
$$C = 2^{11} = 2048 \quad (=2K)$$

- Quelle doit être la longueur d'un mot binaire pour avoir une capacité de représentation $C=64$ kilo ?
 $b=2 \quad l = \ln(C) / \ln(2)$ soit 15,9 bits $\Rightarrow l=16$ bits.

- Si $C = 10$ Méga, quelle doit être la longueur des mots binaires écrits en base 16 (hexadécimal) ?
 $b=16, l=5,8$ soit $l=6$ caractères hexa (ou 3 octets).

I - Représentation des entiers positifs.

L'ensemble des entiers positifs donnés en format fixe est $N=[0,1,2,3,\dots,N_{\max}]$. Sa représentation circulaire est :



Pour une capacité de représentation C donnée l'utilisation de l'opération INC (incrémenter de +1) est possible.

Mais on constate que $N_{\max} + 1 = 0$.

Cela signifie que pour le format considéré, il y a dépassement. Celui-ci étant marqué par un indicateur $V=1$ (Overflow)

- Dans l'utilisation de l'opération DEC (décrémenter), on remarque que $0-1 = N_{\max}$. Il faut alors prendre en compte la retenue dans l'opération suivante ($C=1$, Carry).

Exercice : Montrer que pour $l=8$ en binaire la somme de $A=160_{10}$ (soit $A0_H$) et de $B=127_{10}$ (soit $7F_H$) est 32_{10} et $V=1$.

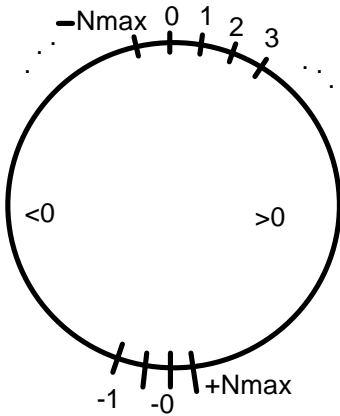
II - Représentation des entiers relatifs.

L'ensemble des entiers relatifs donnés en format fixe est : $N = [-N_{\max}, \dots, -1, 0, 1, \dots, +N_{\max}]$. Il est donc nécessaire de coder le signe algébrique. Pour cela, plusieurs représentations sont possibles.

II-1) représentation par bit de signe et valeur absolue.

Si le nombre s'écrit $N_b = (s a_{n-1} \dots a_1 a_0)_b$ alors s représente le signe et $a_{n-1} \dots a_0$ la valeur absolue. Le bit d'ordre n est alors réservé au signe et vaut 1 si s signe - et 0 si s signe +.

Représentation circulaire :



Cette représentation :
 -nécessite un traitement séparé du signe et de la valeur absolue dans les opérations arithmétiques,
 -possède deux représentations du zéro :
 -0 = 10000000
 +0 = 00000000

Exemples : Si $b=2, l=8 \Rightarrow C=256$.

$$(+13)_{10} = (00001101)_2 = (0D)_H$$

$$(-13)_{10} = (10001101)_2 = (8D)_H$$

$$+N_{\max} = (01111111)_2 = (7F)_H = (+127)_{10}$$

$$-N_{\max} = (11111111)_2 = (FF)_H = (-127)_{10}$$

II-2) Représentation par le complément restreint de N (complément à 1).

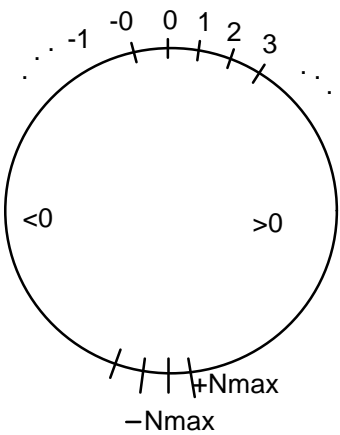
Dans cette représentation, le signe est traité avec la valeur. Il est cependant représenté par le bit de poids fort.

Le complément restreint $C_R N = (2^l - 1)_{10} - N_{10}$ avec $(2^l - 1)$ plus grand nombre que l'on puisse représenter avec le format de l bits.

Il s'obtient donc en inversant chaque bit de N_2 sans oublier le bit de poids fort (signe).

Donc $-N_2$ s'écrit $\overline{N_2}$

Représentation circulaire :



Exemple : Si $l=8$ et $b=2$ $C=256$. Alors

$$N_{\max} = (127)_{10} = (7F)_H$$

- Si $B_{10} = 75$, alors $-75 = C_R B = 255 - 75 = 180$ et $A_{10} = 100$
- Dans ce cas $A - B = (100 - 75) = 100 + 180$, c'est égal à $255 + 25 = 25$ résultat positif.
- $(-75)_{10} = (01001011)_2$; $(-75)_{10} = (10110100)_2 = \overline{N_2}$

Remarques :

- Il y a deux représentations du zéro $+0 : (00)_H$ et $-0 (FF)_H$.
- les instructions INC et DEC ne sont pas utilisables partout.
- Connaître le signe d'un résultat est parfois complexe : par exemple,
 $(-75)_{10} = (180)_{10}$, $(-25)_{10} = (230)_{10}$
 $(-75 - 25) = (180 + 230) = (410)_{10} = (255 + 155) = 155 (> 127)$ d'où $255 - 155 = (100) = \Rightarrow (-100)$.

II-3) Représentation par le complément vrai de N (complément à deux).

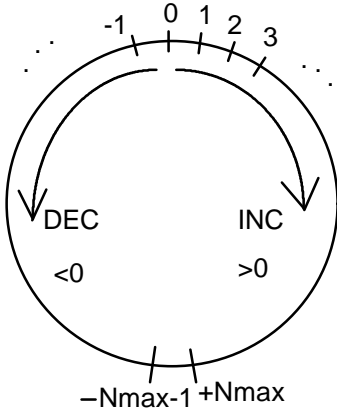
On le définit comme étant la valeur :

$$C_V N = (2^l)_{10} - N_{10} = C_R N + 1$$

Il s'obtient donc en ajoutant 1 au complément restreint. Le premier bit reste toujours le bit de signe.

Donc $-N$ s'écrit $\overline{N} + 1$

Représentation circulaire



Le demi-cercle s'obtient (pour les nombres négatifs) en décalant (INC) d'un pas de façon à superposer +0 et -0. Il y a donc un seul zéro et il est possible d'utiliser les opérateurs INC et DEC.

Exemples de représentation

Valeur algébrique décimale	Représentation des nombres par :		
	Bit de signe et valeur absolue	Complément restreint	Complément vrai
-127	1111 1111	1000 0000	1000 0001
-105	1110 1001	1001 0110	1001 0111
-15	1000 1111	1111 0000	1111 0001
-12	1000 1100	1111 0011	1111 0100
-3	1000 0011	1111 1100	1111 1101
-2	1000 0010	1111 1101	1111 1110
-1	1000 0001	1111 1110	1111 1111
0	0000 0000	1111 1111	0000 0000
+1	0000 0001	0000 0001	0000 0001
+2	0000 0010	0000 0010	0000 0010
+3	0000 0011	0000 0011	0000 0011
+12	0000 1100	0000 1100	0000 1100
+15	0000 1111	0000 1111	0000 1111
+105	0110 1001	0110 1001	0110 1001
+127	0111 1111	0111 1111	0111 1111

Exemples : $N_{10} = 75 \rightarrow (01001011)_2$
 $N_{10} = -75 \rightarrow (10110101)_2$

100	01100100	01100100	
-75	-01001011	+10110101	
=25	00011001	1 00011001	lecture directe du résultat

50	00110010	00110010	
-75	-01001011	+10110101	
-25	11100111	11100111	le résultat est en complément à deux : 256-231=25

Remarques : 1) Pour effectuer une soustraction, il suffit de faire une addition avec le complément à deux. Le résultat se lit directement en complément à deux :

- si le signe est + la lecture est directe,
- si le signe est - on convertit le résultat en recherchant le complément à deux de celui-ci.

2) Il existe une autre méthode pour obtenir le complément vrai. On examine le nombre binaire en commençant par le bit de poids faible. On conserve les zéros s'il y en a jusque et y compris le premier 1 rencontré. On complémente ensuite les autres bits.

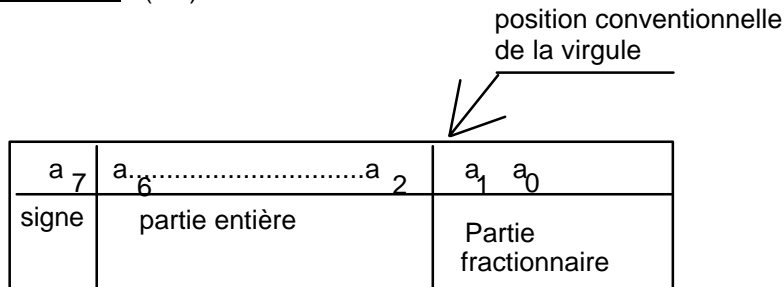
III - Représentation des nombres fractionnaires.

La situation de la virgule dans les nombres fractionnaires se fait sur des conventions suivant sa position à l'intérieur de celui-ci. Elle n'est cependant pas représentée.

III - 1) Virgule placée à un rang fixe quelconque.

Dans cette convention, la virgule est placée de façon immuable.

Exemple de représentation : (l=8)



Exemples :

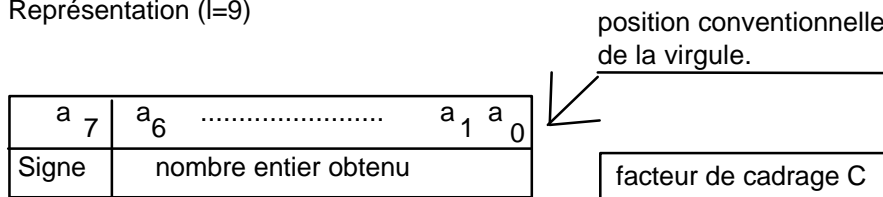
- (+14,75)₁₀ = +(1110,11)₂ -> (00111011)₂
- (-7,5)₁₀ = -(0111,1)₂ -> (1110010)₂=C_vN₂ (le complément vrai est pris sur tous les bits)
- (+20)₁₀ = +(10100)₂ -> (01010000)₂
- (+0,25)₁₀ = +(0,01)₂ -> (00000001)₂

Remarque : Dans cette convention, il existe des limites aux dimensions des parties entière et fractionnaire (ici 5 et 2 bits). D'autre part, on n'utilise pas dans certains cas les bits significatifs.

III - 2) virgule située à droite du dernier rang.

Dans cette convention, on ramène tous les nombres fractionnaires à des entiers en les multipliant par (2₁₀)ⁿ, c'est à dire 10₂ⁿ.

Représentation (l=9)



Le but de cette convention est de ne conserver que la partie significative du nombre. Afin de pouvoir faire la lecture de celui-ci, il est nécessaire d'associer à ce nombre un facteur de cadrage (d'échelle) qui n'est autre que c = n (+, - ou 0).

Exemples : (représentation sur 8 bits)

- (+14,75)₁₀ = +(1110,11)₂ = (111011.(10)₂⁻²)₂ --> (00111011)₂ et C=-2.

Cela signifie que la virgule réelle se trouve 2 rangs à gauche de la virgule conventionnelle.

- (+328)₁₀ = (101001000)₂ = (1010010.10²)₂ --> (01010010)₂ et C=2 : virgule réelle 2 rangs à droite.

- (-0,656)₁₀ --> (+0,656)₁₀ = (0,101001)₂ = (101001.10⁻⁶)₂ donc --> (00101001)₂ ; il vient :

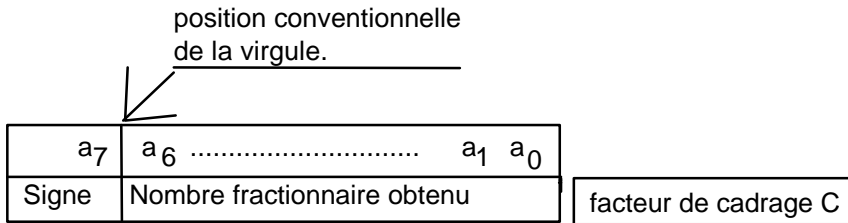
(-0,656)₁₀ --> (11010111)₂ et C=-6 : virgule six rangs à gauche.

Remarque : Le facteur de cadrage peut être modifié pour conserver certains bits significatifs.

III-3) Virgule située à droite du bit de signe.

Dans cette convention, on ramène tous les nombres fractionnaires à un nombre fractionnaire inférieur à un (0,101101....) en les multipliant par 2_{10}^n (10_2^n).

Représentation (l=8)



Le facteur de cadrage est choisi tel que le premier bit derrière la virgule soit 1.

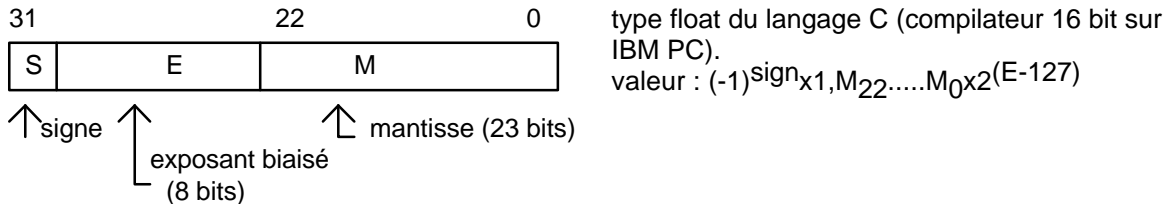
Exemples :

- $(+14,75)_{10} = (1110,11)_2 = (0,111011 \cdot 10^4)_2 \rightarrow (01110110)_2$ et $C=4$. La virgule est située 4 rangs à droite de la virgule conventionnelle.
- $(+328)_{10} = (101001000)_2 = (0,101001 \cdot 10^9)_2 \rightarrow (01010010)_2$ et $C=9$. la virgule est au 9° rang à droite.
- $(-0,3125)_{10} \rightarrow (+0,3125)_{10} = (0,0101)_2 = (0,101 \cdot 10^{-1})_2$ donc $\rightarrow (01010000)_2$ et $C=-1$.
- $(-0,3125)_{10} \rightarrow (1011000)_2$ c'est à dire $(0,11 \cdot 10^{-1})_2$ pour la partie fractionnaire qui se normalise par : $(11100000)_2$ et $C=C-1=-2$ (nouveau facteur de cadrage). La virgule est ici placée 2 rangs à gauche de la virgule conventionnelle.

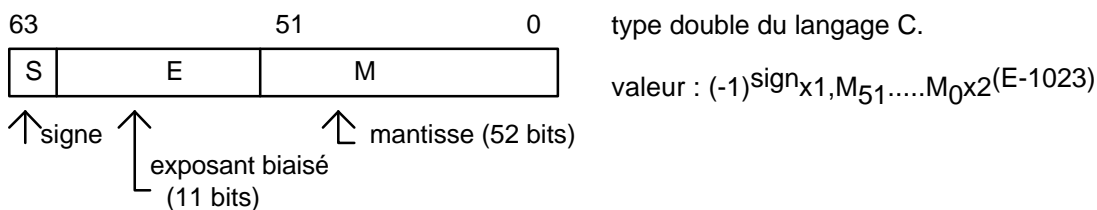
Remarque : Dans cette convention, on peut modifier, si c'est nécessaire, le facteur de cadrage afin d'utiliser au mieux tous les bits significatifs. On dit alors que l'on travaille en virgule flottante

III-4) Norme ANSI IEEE standard 754 pour la représentation des réels.

III-4-a) Simple précision.



III-4-b) Double précision.



IV - Exercices.

- 1) Dans un micro-ordinateur une variable entière simple est représentée sur 2 octets. Quelle est la capacité de représentation ?
- 2) Représenter en binaire sur 8 bits les nombres $(+98)_{10}$ et $(-98)_{10}$ dans les trois conventions de représentation. Les écrire en hexadécimal.

3) Une calculatrice travaille en binaire sur un format de 12 bits avec la convention du complément vrai et donne ses résultats en hexadécimal.

Quelles sont en base 10 les nombres équivalents aux résultats suivants :

$(B46)_{16}$; $(07F)_{16}$; $(FF8)_{16}$.

4) A partir des conventions des nombres entiers et fractionnaires, donner les représentations binaire et hexadécimale dans un format de 8 bits, des nombres $(48,625)_{10}$ puis $-(48,625)_{10}$ représentés par le complément vrai.

5) -a- Pour un système de numération dans la base b de l chiffres de longueur, quelle est la capacité de représentation.

-b- le coût de la représentation est environ : $p=b.l$; pour une capacité C donnée, quelle est la base de représentation la moins coûteuse ?

-c- comparer par rapport à cette base théorique le coût de la base 2, puis 3, 4, 10, 16, pour une capacité de 65536.

6) Pour effectuer des opérations arithmétiques, on utilise la méthode du complément vrai sur des nombres de 2 chiffres s'écrivant D1D0.

a) Le format étant donné, D1D0 est un nombre dont l'écriture est comprise entre 00 et 99. Si $C_V N = 100 - N$, le signe du nombre N dépend de D1. Si $D1 (> \text{ ou } =) 5$ alors $N < 0$; si $D1 < 5$ $N > 0$.

Représenter de -10 à +10 les nombres signés. Quelles sont les valeurs de -Nmax et + Nmax.

b) Effectuer $7+3$; $7-3$; $3-7$; $-3-7$. Expliquer vos résultats.

c) Effectuer $40-25$; $33-48$; $-15-12$; $15-15$.

7) Si on effectue l'addition de 2 nombres signés N1 et N2 de signes respectifs s1 et s2, le résultat possède un signe s3. A partir des états 0 ou 1 des 3 variables s1,s2 et s3, donner l'état du calcul sachant qu'il y a 8 combinaisons possibles.

États possibles : signe réel du résultat : positif N=0, négatif N=1 Débordement V=1

Il y aura donc deux variables d'état N et $V = f(s1,s2,s3)$. Donner les équations de N et V. Proposer un schéma à NANDs.

Cours 3 - Les Codes

I - Définitions

Un code binaire est une convention permettant de traduire une donnée quelconque en une grandeur ne comportant que des 0 et des 1. Il adapte le langage humain au langage de la machine électronique et inversement.

La numérotation binaire, bien connue, est un code permettant de transformer les nombres décimaux en nombres binaires et inversement : c'est le code binaire naturel (CBN). Il convient pour effectuer des opérations arithmétiques sur des nombres à base 10, mais on peut avoir à pratiquer sur des nombres d'autres opérations (mise en mémoire, comptage, transmission), et dans ce cas, le code binaire naturel n'est pas forcément le meilleur procédé à utiliser.

On peut aussi avoir à traiter des données quelconques autres que des nombres (lettres de l'alphabet, ordre de télécommande...). La numérotation binaire est alors carrément impossible.

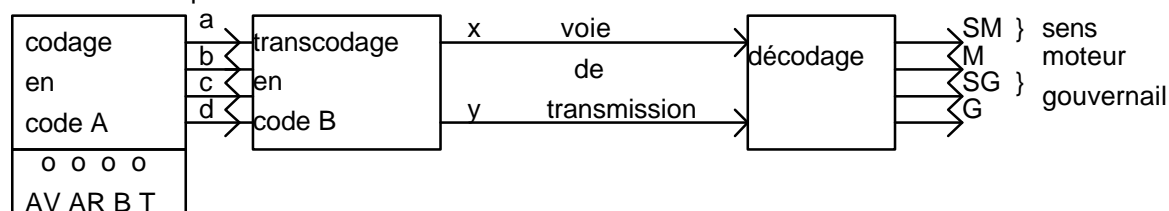
On devra donc utiliser des codes particuliers, ainsi que des opérations permettant de passer d'un code à l'autre.

données (utilisateur clavier)	codage ----->	grandeur binaire
grandeur binaire code 1	transcodage ----->	grandeur binaire code 2
grandeur binaire	décodage ----->	langage compréhensible à l'utilisateur (visu.....)

Exemple : Télécommande d'une maquette de bateau.

On dispose de 4 commandes : marche avant, bâbord, marche arrière, tribord.

Pour simplifier le système de télécommande, on choisit de n'utiliser que 2 bits pour transmettre les ordres à la maquette.



clavier	a b c d	x y	M SM	G SG	
AV	0 0 0 1	0 0	1 1	0 X	M=0 moteur arrêt
AR	0 0 1 0	0 1	1 0	0 X	M=1 moteur marche
B	0 1 0 0	1 0	0 X	1 1	G=0 gouvernail centre
T	1 0 0 0	1 1	0 X	1 0	G=1 bâbord/tribord

En absence d'ordre AV ou AR : M=0
B ou T : G=0

II - Différents types de codes

Il existe un certain nombre de codes qui possèdent chacun leurs particularités et qui correspondent à une application précise.

- codes arithmétiques qui permettent de faire les calculs :
 - codes pondérés,
 - codes décimaux,
 - codes auto-complémentés.
- codes alphanumériques qui n'ont aucune propriété arithmétique mais qui servent à représenter des lettres, des chiffres, des signes typographiques. Code télégraphique international ASCII (American Standard Code for Information Interchange).
- codes de position mécanique {code adjacent sans régime transitoire parasite }
- codes de transmission
 - codes détecteurs d'erreur,
 - codes autocorrecteurs.

II-1) Codes arithmétiques

* codes pondérés : chaque bit de ces codes représente un poids dont l'équivalent en base 10 de chaque combinaison est donné par la somme des poids de tous les bits de la combinaison égaux à 1. Chacune des combinaisons à un équivalent décimal, supérieur d'une unité à celui de la combinaison précédente.

* codes décimaux : ce sont des codes à 10 combinaisons représentant les 10 chiffres décimaux. Ils comportent au moins 4 bits et ils sont par conséquent redondants (toutes les combinaisons binaires ne sont pas utilisées).

Exemple :

décimal	code binaire naturel 8421 (poids)	Décimal Code Binaire (DCB) plutôt appelé Binary Coded Decimal (BCD) (poids) 8421
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	0101
6	0110	0110
7	0111	0111
8	1000	1000
9	1001	1001
10	1010	Non codé
11	1011	Non codé
12	1100	Non codé
13	1101	Non codé
14	1110	Non codé
15	1111	Non codé

Les codes décimaux servent à l'affichage ou à l'impression des nombres.

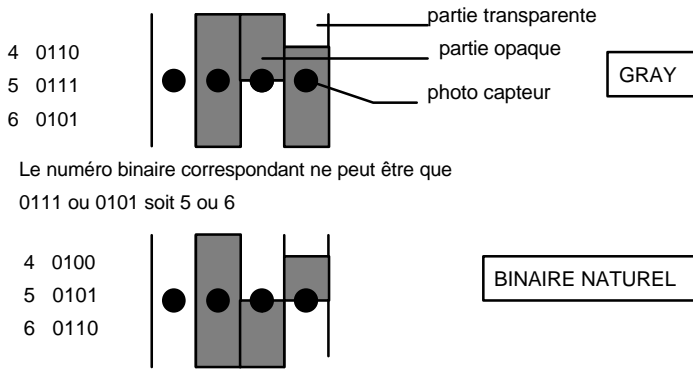
Pour entrer ou sortir en machine un nombre décimal, on le transcode en passant par le code BCD.

$(172)_{10} \text{ -----} \rightarrow (0001\ 0111\ 0010)_{BCD} \text{ -----} \rightarrow (10101100)_2$

L'arithmétique en code BCD est compliquée (voir addition plus loin dans le poly).

II-2) Codes de position : (réfléchis ou reflex)

Une position angulaire θ analogique est transformée en grandeur numérique au moyen d'un disque codé lié à la pièce dont on veut repérer le mouvement. Le disque est divisé en p couronnes correspondant à p bits et repérant ainsi 2^p positions ; chaque couronne présente des parties soit opaques et transparentes (lecture optique) soit conductrices et isolantes (lecture électrique). A chaque position du disque correspond alors un nombre binaire.



On peut bien sûr coder ainsi un déplacement linéaire avec une plaque formée de pistes parallèles.

Le code le plus utilisé est le code Gray. Il existe aussi des codes (BDR . BDR xs 3) où les chiffres décimaux sont codés séparément et juxtaposés (code qui se prête mieux à l'affichage).

Dans ces codes, un seul bit change d'état à la fois quand on passe d'une combinaison à la suivante, de façon à limiter les erreurs.

En binaire naturel on obtient 0100 ou 0101 ou 0110 ou 0111 soit 4 ou 5 ou 6 ou 7.

Décimal	Gray	Bin. Dec. Refl.	BDR xs 3
0	0000	0000 0000	0010 0010
1	0001	0000 0001	0010 0110
2	0011	0000 0011	0010 0111
3	0010	0000 0010	0010 0101
4	0110	0000 0110	0010 0100
5	0111	0000 0111	0010 1100
6	0101	0000 0101	0010 1101
7	0100	0000 0100	0010 1111
8	1100	0000 1100	0010 1110
9	1101	0000 1101	0010 1010
10	1111	0001 1101	0110 1010
11	1110	0001 1100	0110 1110
12	1010	0001 0100	0110 1111
13	1011	0001 0101	0110 1101
14	1001	0001 0111	0110 1100
15	1000	0001 0110	0110 0100
16	-----	0001 0010	0110 0101
17		0001 0011	0110 0111
18		0001 0001	0110 0110
19		0001 0000	0110 0010
20		0011 0000	0111 0110

Ces codes ne sont pas pondérés. Ils sont cycliques si un seul bit change d'état entre la dernière combinaison et la première. (Les codeurs angulaires sont obligatoirement cycliques).

II-3) Codes détecteurs

Dans la transmission d'une grandeur numérique peuvent se glisser des erreurs (1 transformé en 0 ou 0 en 1) quelque soit le moyen de transmission (bande magnétique, carte perforée, procédés optiques) S'il peut se produire une erreur sur un bit (probabilité d'erreur p : ex $1/10000$; $p = 10^{-4}$), il est beaucoup plus rare que deux erreurs se produisent simultanément (probabilité p^2).

En partant de ce principe, on utilisera pour transmettre les données, des codes dont toutes les combinaisons ont une caractéristique commune :

- chaque combinaison ne comprend par exemple que 2 bits égaux à un et tous les autres à zéro.
- chaque combinaison contient un nombre pair de 1.

Les combinaisons reçues qui n'ont pas cette caractéristique n'appartiennent pas au code et sont donc erronées.

Ces codes sont tous redondants.

Exemples :

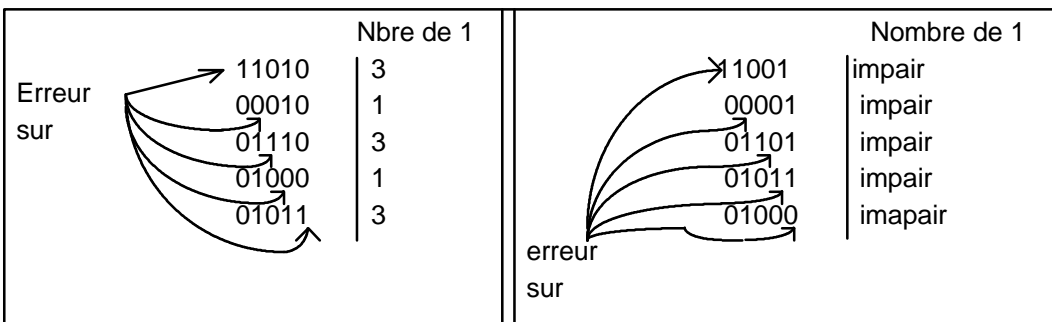
code "2 parmi 5 "

code BCD + bit de parité

décimal, pondéré (sauf 0) redondance : 22 (32 combinaisons. 10 utiles) On ajoute 1 cinquième bit de telle sorte que chaque combinaison contienne un nombre pair de 1.
 63210 ← poids 8421 0 ← poids

0	00110	0000 0	(dernière colonne : bit de parité)
1	00011	0001 1	
2	00101	0010 1	
3	01001	0011 0	
4	01010	0100 1	
5	01100	0101 0	
6	10001	0110 0	
7	10010	0111 1	
8	10100	1000 1	
9	11000	1001 0	

Soit à transmettre $(4)_{10}$ dans les 2 codes précédents : $(4)_{10}$ 01010 $(4)_{10}$ 01001
 Quel que soit le bit sur lequel se produit l'erreur, la combinaison n'appartient pas au code.



A la réception du code, on vérifie la parité. Si elle est correcte le mot est validé. (Voir TP : codeur de Hamming).

Codes autocorrecteurs :

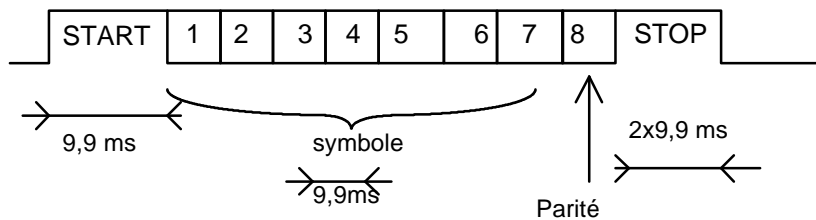
Quand on sait qu'on a une erreur dans un mot, il est intéressant de la localiser pour pouvoir la corriger : il suffit alors d'inverser le bit fautif). On obtient ainsi une transmission parfaite. On construit ces codes en ajoutant aux bits d'informations des bits de "contrôle".

II-4)Codes alphanumériques :

Ils n'ont aucune propriété arithmétique mais ils servent à coder des chiffres, des lettres et des signes. On travaille par comparaison et non par calcul.

Exemple : code télégraphique international N°5 (dit code ASCII)

Chaque symbole y est représenté par 8 bits (dont 1 de parité) précédés d'un signal de début (START) et d'un signal de fin (STOP).



Ce code est utilisé par les réseaux informatiques assurant les connexions entre les ordinateurs et les organes périphériques (claviers, visu., imprimantes).

Le Baud étant l'inverse de la durée d'un signal élémentaire, le code ASCII fonctionne à une vitesse de $1/0,0099 = 110$ bauds.

III - Exercice

Construire un code décimal pondéré 6421.

Cours 4 : Le langage ABEL

I - Le langage ABEL (spécifications)

I-1) Caractères valides

a -z lettres minuscules
 A - Z lettres majuscules
 0 - 9 chiffres
 <espace>
 <tabulation>

!	@	#	\$?	+	&	*	()	-
_	=	[{	}]	;	:	'	"	+
~	\		,	<	>	.	/	^	%	

I-2) Identificateurs

Ce sont des noms qui identifient les composants, les broches les signaux d'entrée et de sortie... Il peuvent avoir jusqu'à 31 caractères de long. Il existe comme dans tout langage des identificateurs réservés ou mots-clefs :

case	goto	property	declarations	if
state	device	in (obsolete)	state_diagram	else
istype	test_vectors	enable (obsolete)	library	then
end	macro	title	endcase	module
trace	endwith	node	truth_table	equations
options	when	flag (obsolete)	pin	with
fuses				

I-3) Constantes

Constantes	description
.C.	entrée d'horloge bas-haut-bas (monostable)
.D.	front descendant d'une entrée horloge
.F.	signal d'entrée ou de sortie flottant
.K.	entrée d'horloge haut-bas-haut (monostable)
.P.	préchargement dans un registre
.SVn	n variant de 2 à 9. Commande l'entrée pour une tension de 2 à 9
.U.	front d'horloge montant
.X.	valeur indéterminée à calculer
.Z.	valeur trois état.

I-4) Blocs

Ce sont des caractères ASCII se trouvant entre des accolades : { ceci est un bloc }

I-5) Commentaires

Les commentaires commencent par des guillemets et finissent soit par des autres guillemets soit par une fin de ligne.

I-6) Nombres

Toutes les opérations en ABEL invoquant des valeurs numériques sont faites avec une précision de 32 bits.

Nom de la base	base	symbole
binary	2	^b
octal	8	^o

decimal	10	^d (par défaut)
hexadecimal	16	^h

I-7) Chaînes de caractères

Les chaînes de caractères sont des caractères ASCII entourés par des apostrophes :
'ceci est une chaîne'

I-8) Opérateurs expressions et équations

Les opérateurs logiques sont :

Opérateur	description	priorité
!	non ou complément à 1	1
&	et	2
#	ou	3
\$	ou exclusif	3
!\$	identité (non ou exclusif)	3

Les opérateurs arithmétiques peuvent permettre de définir des opérations arithmétiques entre plusieurs membres d'une expression.

Opérateur	exemple	description	priorité
-	-A	négation ou complément à deux	1
-	A-B	soustraction	3
+	A+B	addition	3
*	A*B	multiplication	2
/	A/B	division entière non signée	2
%	A%B	reste de la division (opérateur modulo)	2
<<	A<<B	décalage gauche de A de B bits	2
>>	A>>B	décalage droit de A de B bits.	2

Les opérateurs relationnels sont :

opérateur	description	priorité
==	égalité	4
!=	différent	4
<	inférieur	4
<=	inférieur ou égal	4
>	supérieur	4
>=	supérieur ou égal	4

Les opérateurs d'affectation sont de deux sortes :

= affectation combinatoire

:= affectation séquentielle.

Tous les opérateurs sont associatifs gauche-droite. On peut utiliser des parenthèses pour changer la priorité.

I-9) Les ensembles

Une ensemble est une collection de signaux et constantes qui opèrent comme un seul. Un ensemble est représenté comme une liste de signaux et constantes séparées par des virgules et entourés par des crochets.

[B7,B6,B5,B4,B3,B2,B1,B0] est un ensemble de huit signaux.

On peut réaliser des opérations sur les ensembles à condition qu'ils aient le même nombre d'éléments.

Exemples :

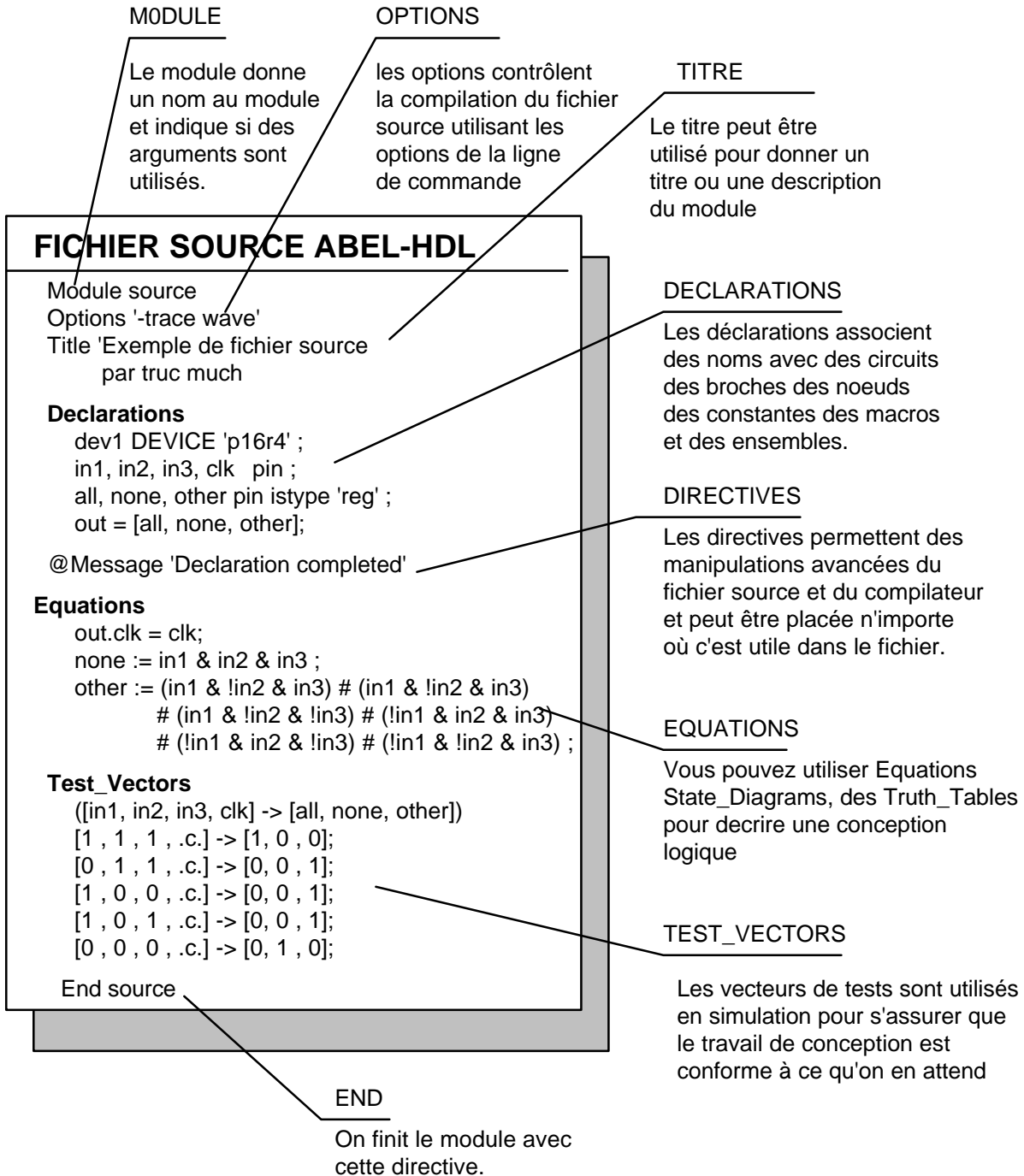
Addr = [A15,A14,A13]; "déclaration d'ensemble (A13 est considéré comme le poids faible)

chip_sel = Addr==[1,0,1]; est équivalent à chip_sel=A15 & !A14 & A13 ou chip_sel = Addr==5;

Il existe une déclaration simplifiée lorsque les broches sont numérotées :

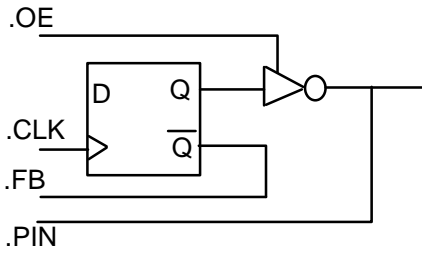
Addr = [a15..a0] déclare 16 entrées de a0 à a15.

I-10) La structure de programme

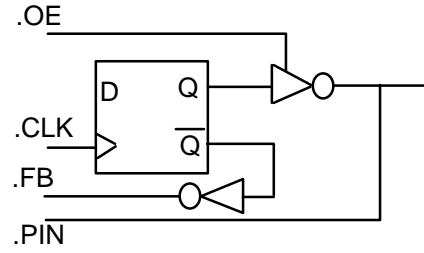


I-11) Les points extensions

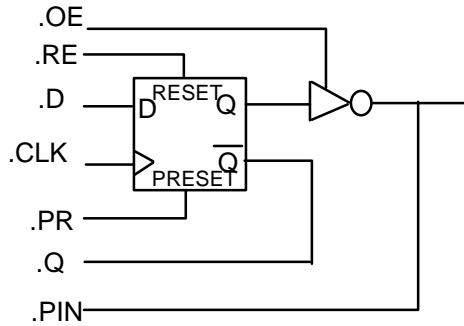
Il s'agit de mettre un caractère point suivi d'une extension qui a une signification déterminée. On donne quelques unes de ces extensions pour la bascule D la bascule RS et la bascule JK ci-après. Celles de la D Latch et de la bascule T ne seront pas détaillées.



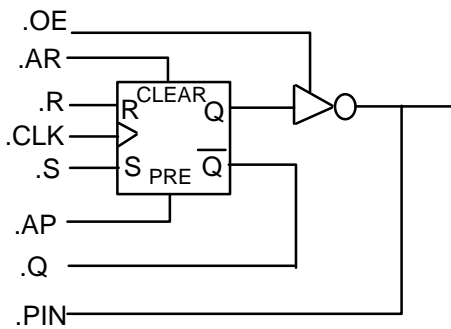
Point-extensions dans une architecture inversée



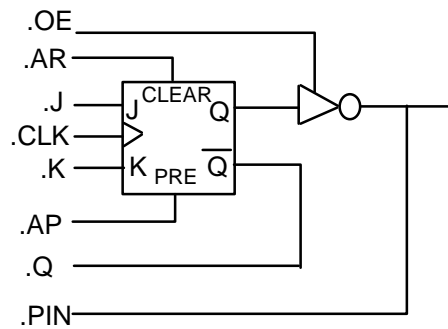
Point-extensions dans une architecture non inversée



Point-extensions détaillées
Pour une architecture en
bascule D



Point-extensions détaillées
Pour une architecture en
bascule RS



Point-extensions détaillées
Pour une architecture en
bascule JK

I-12) Les macros

Des exemples seront plus parlant qu'un grand discours.

Nand3 MACRO (A,B,C) {!(?A & ?B & ?C)} ; s'utilise comme : D = Nand3(Clock,Helle,Busy) ;

syntaxe : id_macro MACRO [(arg_faux [,arg_faux]...)] { bloc } ;

Remarque : une macro dans un langage est toujours du remplacement de texte par du texte (et ceci avant la compilation). Cela a des conséquences sur les priorités. Par exemple :

Y1 macro { B # C } alors X1 = A & Y1 s'interprétera A & B # C donc (A & B) # C (ce qui peut surprendre !)

II) Applications

Nous donnons un ensemble de programmes destinés à la compréhension des notions que nous venons d'exposer.

II-1) Passage table de vérité -> équations

La compilation de ce fichier exo1.abl que l'on vous demande de réaliser donne :
 $out = (B \& C \# !A \& B)$ que l'on interprète

comme : $out = B.C + \bar{A}.B$

" : est le début d'un commentaire qui se termine en fin de ligne
 istype 'com' : signifie sortie combinatoire

II-2) Simulation

Notre but est de comprendre comment sont interprétées les lignes de la table de vérité non spécifiées. Nous avons 3 entrées, il devrait donc y avoir 8 spécifications ($8 = 2^3$). La partie test_vectors est destinée à la simulation : on lui demande de simuler le résultat pour toutes les possibilités sur les entrées (8 lignes) et le .X. signifie qu'on lui demande de calculer la sortie. Il est possible de mettre des 0 ou des 1 à la place des .X. pour vérifier un résultat.

II-3) Introduction des équations logiques

On programme maintenant directement des équations logiques avec les opérateurs :

- NON : ! (priorité 1)
- ET : & (priorité 2)
- OU : # (priorité 3)
- OU EXCLUSIF : \$ (priorité 3)
- IDENTITE : !\$ (priorité 3)
- affectation combinatoire : =

Une compilation pourra simplifier ces équations.

II-4) Plusieurs sorties.

Nous avons amélioré notre manière de programmer de différentes manières :

- définition de nom : X et Xs
- spécification d'une entrée par des chiffres en base 10. On peut faire de même pour les sorties et on peut aussi changer de base avec ^h (hexadécimal), ^b (binaire), ^o (octal), ^d (décimal). Les vecteurs tests peuvent aussi être spécifiés avec des chiffres.

II-5) Méthode SI-ALORS

(voir TD 5)

```
MODULE EXO1
title 'exercice 1 par LeProf'
"entrées
A,B,C pin;
"sortie
out pin istype 'com';
truth_table
([A, B, C] -> out)
[0, 1, 0] -> 1;
[0, 1, 1] -> 1;
[1, 1, 1] -> 1;
[1, 1, 0] -> 0;
end
```

```
MODULE EXO2
title 'exercice 2'
A,B,C pin; "entrées
out pin istype 'com'; "sortie
truth_table ([A, B, C] -> out)
[0, 1, 0] -> 1;
[0, 1, 1] -> 1;
[1, 1, 1] -> 1;
[1, 1, 0] -> 0;
test_vectors ([A, B, C] ->out)
[0, 0, 0] -> .X.;
[0, 0, 1] -> .X.;
[0, 1, 0] -> .X.;
[0, 1, 1] -> .X.;
[1, 0, 0] -> .X.;
[1, 0, 1] -> .X.;
[1, 1, 0] -> .X.;
[1, 1, 1] -> .X.;
end
```

```
MODULE EXO4
title 'exercice 4'
A,B,C pin; "entrées
out pin istype 'com'; "sortie
equations
out = !A&B&C # A&B&C;
test_vectors ([A, B, C] ->out)
[0, 0, 0] -> .X.;
[0, 0, 1] -> .X.;
[0, 1, 0] -> .X.;
[0, 1, 1] -> .X.;
[1, 0, 0] -> .X.;
[1, 0, 1] -> .X.;
[1, 1, 0] -> .X.;
[1, 1, 1] -> .X.;
end
```

```
MODULE EXO5
title 'exercice 5'
A,B,C pin; "entrées
out1,out2 pin istype 'com'; "sortie
truth_table ([A, B, C] -> [out1, out2])
0 -> [ 1 , 0 ]; "entrées spécifiées
2 -> [ 1 , 1 ]; "par des chiffres
4 -> [ 0 , 1 ]; " en base 10
7 -> [ 0 , 0 ];
test_vectors ([A, B, C] ->[out1, out2])
[0, 0, 0] -> .X.;
[0, 0, 1] -> .X.;
[0, 1, 0] -> .X.;
" ajoutez vos vecteurs tests ici
[1, 1, 1] -> .X.;
end
```

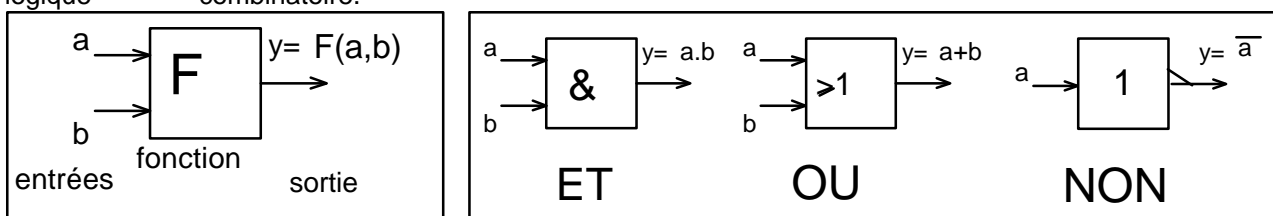
TD1 - Logique et algèbre de Boole.

La logique utilise les lois de l'algèbre de Boole, philosophe et mathématicien anglais (1815-1864). L'algèbre de Boole est applicable à l'étude des systèmes binaires, c'est à dire possédant deux états s'excluant mutuellement : c'est le cas des circuits logiques, base des systèmes numériques. L'étude de ces systèmes (analyse et synthèse des machines numériques) est l'objectif premier de ce cours.

I - Définitions.

- Les états logiques sont représentés par $\{0,1\}$ ou par une réponse en tout ou rien $\{\text{FALSE}, \text{TRUE}\}$
- Une variable booléenne (ou variable logique) est une grandeur, représentée par un symbole qui peut prendre les valeurs 0 et 1 suivant certaines conditions.
- Une fonction logique est représentée par des groupes de variables reliées par des opérateurs logiques : il existe trois opérateurs élémentaires (réunion : OU (OR), intersection : ET (AND) &, complément : NON (NOT)).

Les variables de sortie sont fonction des variables d'entrée et sont booléennes, nous parlerons de logique combinatoire.



II - Représentation des fonctions logiques

- Table de vérité : Une fonction logique sera définie par sa table de vérité. C'est un tableau comportant les valeurs des variables d'entrée et faisant correspondre celle de la fonction (variable de sortie). Elle donne tous les cas de situation d'entrée. Pour éviter les oublis, les combinaisons des valeurs logiques sont données dans l'ordre binaire (Code Binaire Naturel). Deux fonctions qui ont la même table de vérité sont identiques.

- Diagramme de Wenn : Chaque variables logiques divisent l'espace en deux sous-espaces : celui où la variable est vraie (1) et son complément, celui où elle est fausse (0). Ce type de diagramme sera peu employé.

- Diagramme de Karnaugh : c'est un tableau dérivant du diagramme de Wenn et de la table de vérité. Chaque ligne de la table de vérité est représentée par une case dont les coordonnées (lignes, colonne) sont des combinaisons de variables d'entrée.

	a b	y
0	0 0	0
1	0 1	0
2	1 0	0
3	1 1	1

Table de vérité

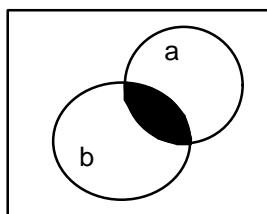


Diagramme de Wenn

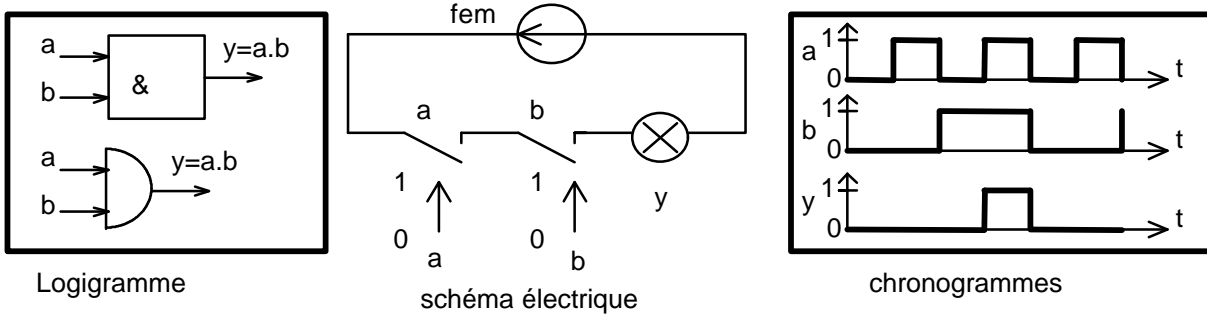
a \ b	0	1
0	0	0
1	0	1

y=F(a,b)
y=a.b

Diagramme de Karnaugh
Tableau de Karnaugh

- Logigrammes : ce sont des schémas logiques représentant les fonctions désirées, ils utilisent des opérateurs élémentaires (ET, OU, NON, ET-NON, OU-NON, OU-EXCLUSIF), ils sont réalisables physiquement avec des circuits électriques, électroniques ou pneumatiques plus ou moins complexes, c'est la concrétisation des machines numériques.

- Chronogrammes : ce sont des graphiques représentant l'évolution des variables d'entrée et de sortie (signaux logiques) en fonction du temps. La bonne interprétation de ces signaux est fondamentale pour le technicien qui disposera d'un oscilloscope à la mise au point du montage.

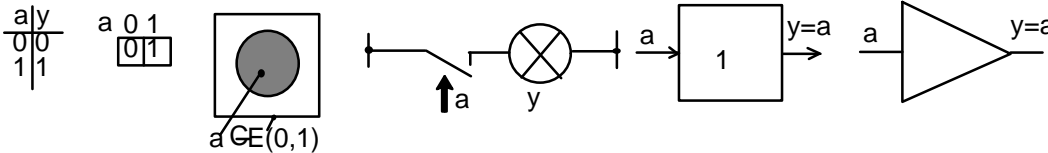


III - Fonctions élémentaires.

a, b, y appartiennent à {0,1}, y=f(a,b)

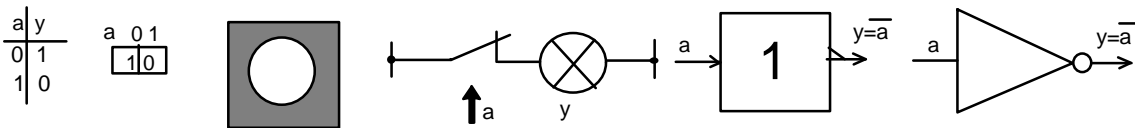
III-1) Fonction d'une variable

- Fonction identité ou OUI : y=a

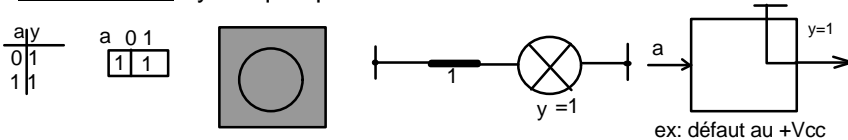


- Fonction complément ou NON : y = /a (se lit " a barre") (noté y=!a en langage ABEL)

(noté le plus souvent a avec une barre par dessus \bar{a})

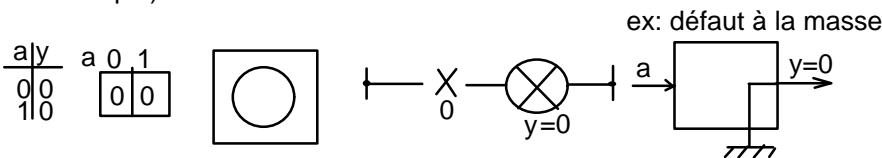


- Fonction vrai: y=1 quelque soit a.



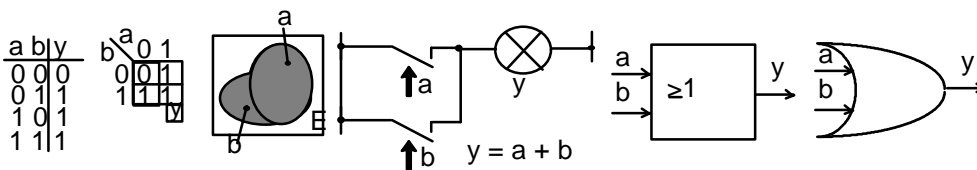
- Fonction faux: y=0 quelque soit a.

(les deux fonctions vrai et faux existent dans les circuits programmables UAL, car elles ont un sens en arithmétique).



III-2) Fonction de deux variables : f(a,b)

- Fonction OU: ("addition" logique, réunion) y= a+b =a U b =a V b (y=a#b en ABEL)



à compléter en cours:

- élément neutre

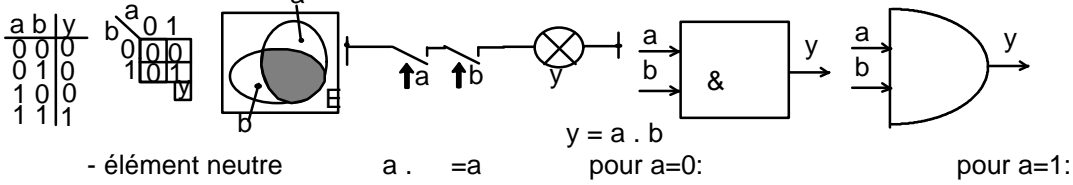
a + =a

pour a=0:

pour a=1:

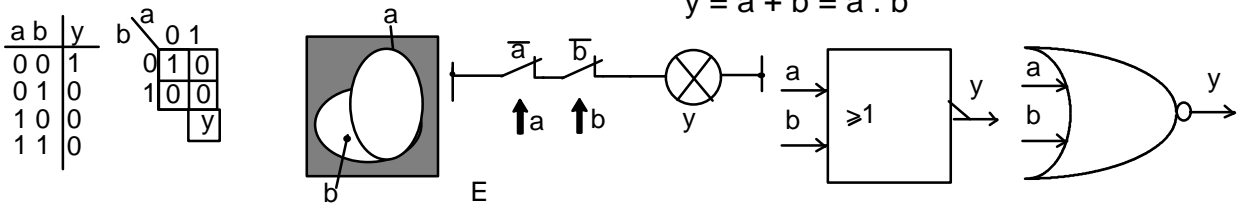
- élément absorbant $a + 1 = 1$
- idempotence $a + a = a$
- complément $a + \bar{a} = 1$
- commutativité $a + b = b + a$

- **Fonction ET:** ("produit" logique, intersection) $y = a \cdot b$ ($y = a \& b$ en ABEL)
 $y = A \cap B = A \wedge B$

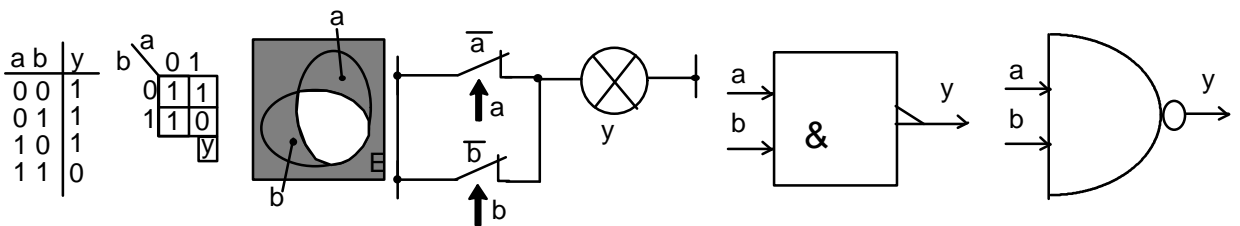


- élément neutre $a \cdot 1 = a$
- élément absorbant $a \cdot 0 = 0$
- idempotence $a \cdot a = a$
- complément $a \cdot \bar{a} = 0$
- commutativité $a \cdot b = b \cdot a$

- **Fonction OU-NON:** (NOR) (NI) $y = \overline{a + b}$ en ABEL
 $y = \overline{a + b} = \bar{a} \cdot \bar{b}$

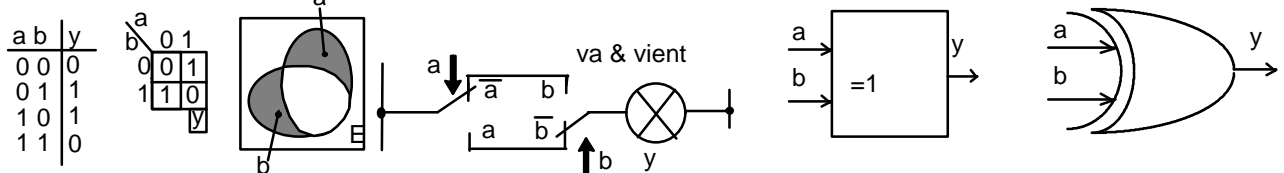


- **Fonction ET-NON:** (NAND) (ON) $y = \overline{a \cdot b}$ en ABEL
 $y = \overline{a \cdot b} = \bar{a} + \bar{b}$



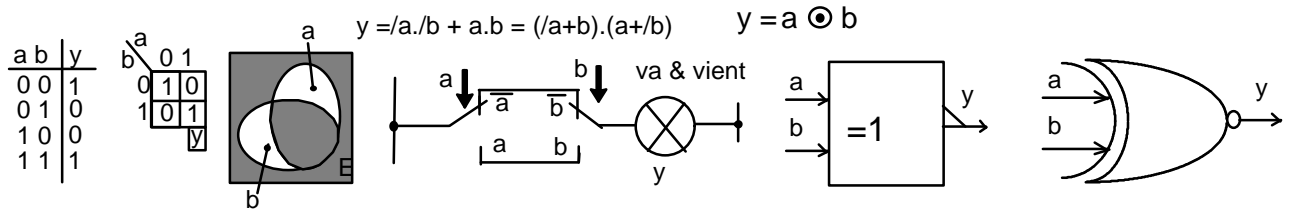
Les opérateurs OU-NON et ET-NON sont très importants, car ce sont des opérateurs complets, ils permettent la synthèse de toutes les fonctions à eux seuls, ce n'est pas le cas du ET ni du OU.

- **Fonction OU-EXCLUSIF:** (X-OR) $y = a \oplus b$ ($y = a \$ b$ en ABEL)
 $y = \bar{a} \cdot b + a \cdot \bar{b} = (a + b) \cdot \overline{(a + b)}$



Le OU-EXCLUSIF joue un rôle très important en arithmétique (1/2 additionneur).

- **Fonction IDENTITE:** c'est le complément du OU-EXCLUSIF, son emploi en temps qu'opérateur est rare. Il est noté $y = a !\$ b$ en langage ABEL.



- Les 16 fonctions à 2 variables d'entrée: $y=f(a,b)$ (voir TD) :

1, 0, a, /a, b, /b, a.b, /a.b, a./b, /a./b, a+b, /a+b, a+/b, /a+/b, $a \oplus b$, $/(a \oplus b)$

IV - Exercice : les 16 fonctions F(a,b)

		variables d'entrées		Intersection (ET)				Non		Dilemmes	
n	b	a		1	2	3	4	5	6	7	8
0	0	0		1	0	0	0	1	1	1	0
1	0	1		0	1	0	0	1	0	0	1
2	1	0		0	0	1	0	0	1	0	1
3	1	1		0	0	0	1	0	0	1	0
DK	0	0	1								
	1										
Expression F											
Complément /F											
symbole Ansi											
Equivalent d'après /F											
symbole normalisé											

		variables d'entrées		Réunion (OU)				Oui		Permanent	
n	b	a		9	10	11	12	13	14	15	16
0	0	0		1	1	1	0	0	0	1	0
1	0	1		1	1	0	1	1	0	1	0
2	1	0		1	0	1	1	0	1	1	0
3	1	1		0	1	1	1	1	1	1	0
DK	0	0	1								
	1										
Expression F											
Complément /F											
symbole Ansi											
Equivalent d'après /F											
symbole normalisé											

TD 2 - Les fonctions Booléennes.

Le but ultime de la logique est de matérialiser à l'aide de composants des fonctions booléennes. Avant d'apprendre à le faire, il nous faut apprendre à manipuler ces fonctions.

I - Définitions.

Une fonction booléenne est une fonction qui à un ensemble de variables d'entrées booléennes fait correspondre une variable booléenne. Elle se représente en général comme une association de sommes (ou logique ou conjonction) et de produits (et logique ou disjonction).

Si l'expression est une somme de produits, la forme est dite disjonctive. Par exemple :
 $a.b.d+a./b+b.c$ (noté !a & b & d # a & !b # b & c en langage ABEL)

Si l'expression est un produit de sommes, le forme est dite conjonctive. Par exemple :
 $(/a+c+d).(a+b).(/c+d)$ (noté (!a#c#!d)&(a#b)&(!c#d) en ABEL)

Une fonction booléenne est dite sous forme normale ou canonique si chaque terme contient toutes les variables.

$/a./b./c+a.b.c+a./b.c$: est sous forme normale disjonctive,
 $(/a+/b+/c).(a+b+c).(a+b+c)$: est sous forme normale conjonctive.

Lorsqu'une fonction booléenne n'est pas sous forme normale, elle est dite sous sa forme simplifiée.

II - Les représentations.

II-1) Table de vérité .

Cette représentation a été définie précédemment, elle suffit à définir complètement la fonction à réaliser. Si nous ne définissons que les cas où la fonction est vraie, implicitement les combinaisons manquantes seront celles où la fonction est fausse et réciproquement. Pour les besoins d'un automate une table peut ne pas être complète, les combinaisons non utilisées pourront être remplacées par 0 ou 1 au choix du concepteur en vue d'une meilleure simplification. Ces cas seront repérés par la lettre phi (ϕ) ou par la lettre X.

n	c b a	f	
0	0 0 0	0	y0
1	0 0 1	1	y1
2	0 1 0	1	y2
3	0 1 1	1	y3
4	1 0 0	0	y4
5	1 0 1	0	y5
6	1 1 0	1	y6
7	1 1 1	1	y7

$f = y1+y2+y3+y6+y7$

$f = \{1,2,3,6,7\}_1$

$\bar{f} = \{0,4,5\}_0$

\triangleright

$n=cba = a.1+b.2+c.4$

Remarque :

En langage ABEL une table de vérité s'écrit comme ci-contre. Il s'agit ici d'une fonction de 3 variables d'entrées A, B et C et d'une variable de sortie : out.

```
truth_table
([A, B, C] -> out)
[0, 1, 0] -> 1;
[0, 1, 1] -> 1;
[1, 1, 1] -> 1;
[1, 1, 0] -> 0;
```

II -2) Représentation numérique.

Si nous affectons à chaque variables un poids binaire, nous effectuerons la réunion des combinaisons où $F=1$, nous écrirons $F = \{1,2,3,6,7\}_1$, nous pouvons aussi définir le complément $/F = \{0,4,5\}_0$.

II-3) Changement de représentation.

* Lecture des 1 de la table de vérité : nous obtenons la première forme canonique.
 Cette présentation de l'équation est bien adaptée à la synthèse avec des ET/OU ou des ET-Non.

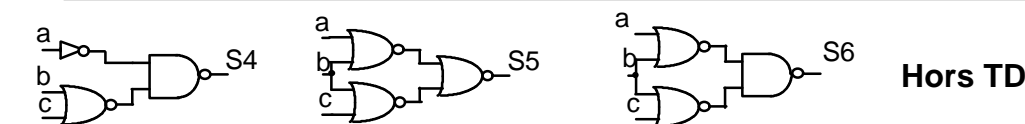
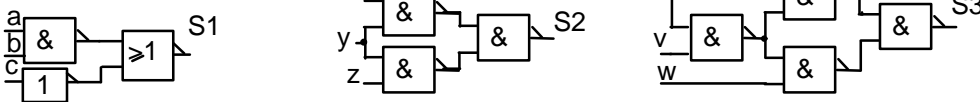
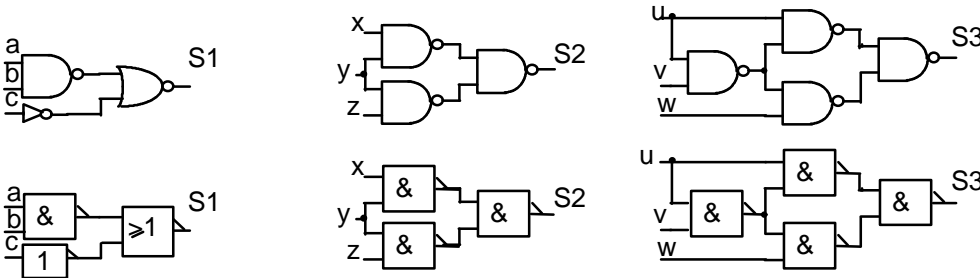
$f = y_1 + y_2 + y_3 + y_6 + y_7$ $f = \bar{c}.b.a + \bar{c}.b.\bar{a} + \bar{c}.b.a + c.b.\bar{a} + c.b.a$	$\Sigma \Pi$
--	--------------

* Lecture des 0 de la table de vérité : nous obtenons la deuxième forme canonique.
 Cette présentation de l'équation est bien adaptée à la synthèse à OU/ET ou OU-Non.

$\bar{f} = \{0,4,5\}_0$ $\bar{f} = y_0 + y_4 + y_5 \rightarrow \overline{f = y_0 + y_4 + y_5} = \bar{y}_0 . \bar{y}_4 . \bar{y}_5$ $f = \bar{c}.\bar{b}.\bar{a} . \bar{c}.\bar{b}.\bar{a} . \bar{c}.\bar{b}.a$ $f = (c+b+a).(\bar{c}+\bar{b}+\bar{a}).(\bar{c}+\bar{b}+a)$	$\Pi \Sigma$
---	--------------

III - Exercices.

1°) Donnez les expressions booléennes représentées par :



Hors TD

Ecrire ensuite ces expressions en utilisant la syntaxe ABEL.

2°) Exprimer les équations suivantes sous la forme canonique disjonctive (somme de produits) :

$y_1 = a+b,$
 $y_2 = a.b.c + /a.b$

3°) Même question, mais sous forme conjonctive (produit de somme) :

$y_3 = a+b+/a.b.c,$
 $y_4 = (a+b+c).(+/a+b).$

Indication : on peut utiliser les zéros de la table de vérité.

4°) Soient les trois fonctions booléennes :

$F_1 = (/a.b+a./b)./c + (/a./b+a.b).c$
 $F_2 = a./b + b./c + c./a + a.b.c$
 $F_3 = (a+b+c).(a+b+/c).(a+/b+c).(a+/b+c).(+/a+b+c).$

- a) Ecrire les tables de vérité correspondantes (dont une avec la syntaxe ABEL).
- b) Ecrire ces fonctions sous formes canoniques disjonctives et conjonctives.
- c) Donner les représentations numériques de ces fonctions.

TD 3 - Simplification des fonctions logiques.

Les formes canoniques ne sont pas adaptées à la synthèse directe des fonctions, seules quelques technologies (réseaux programmables, ...) utilisent ces présentations. Il faut simplifier ces fonctions pour en obtenir une forme minimale et donc minimiser les coûts de production.

Pour simplifier une expression, il existe des méthodes algébriques et des méthodes graphiques.

I - Tableaux de Karnaugh.

L'emploi de ce tableau est possible jusqu'à six variables, au delà il ne reste que les méthodes algébriques, c'est un outil puissant et c'est souvent le plus rapide.

<table border="1" style="width: 100%; border-collapse: collapse;"> <tr><th>b a</th><th>f</th></tr> <tr><td>0 0</td><td>y0</td></tr> <tr><td>0 1</td><td>y1</td></tr> <tr><td>1 1</td><td>y3</td></tr> <tr><td>1 0</td><td>y2</td></tr> </table> <div style="margin-top: 10px;"> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr><th>a</th><th>0</th><th>1</th></tr> <tr><th>b</th><td>$\overline{a.b}$ y0</td><td>$a.b$ y1</td></tr> <tr><th>1</th><td>$\overline{a.b}$ y2</td><td>$a.b$ y3</td></tr> </table> <p>2 variables: 4 cases</p> </div>	b a	f	0 0	y0	0 1	y1	1 1	y3	1 0	y2	a	0	1	b	$\overline{a.b}$ y0	$a.b$ y1	1	$\overline{a.b}$ y2	$a.b$ y3	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr><th>d c b a</th><th>f</th></tr> <tr><td>0 0 0 0</td><td>y0</td></tr> <tr><td>0 0 0 1</td><td>y1</td></tr> <tr><td>0 0 1 0</td><td>y2</td></tr> <tr><td>0 0 1 1</td><td>y3</td></tr> <tr><td>0 1 0 0</td><td>y4</td></tr> <tr><td>0 1 0 1</td><td>y5</td></tr> <tr><td>0 1 1 0</td><td>y6</td></tr> <tr><td>0 1 1 1</td><td>y7</td></tr> <tr><td>1 0 0 0</td><td>y8</td></tr> <tr><td>1 0 0 1</td><td>y9</td></tr> <tr><td>1 0 1 0</td><td>yA</td></tr> <tr><td>1 0 1 1</td><td>yB</td></tr> <tr><td>1 1 0 0</td><td>yC</td></tr> <tr><td>1 1 0 1</td><td>yD</td></tr> <tr><td>1 1 1 0</td><td>yE</td></tr> <tr><td>1 1 1 1</td><td>yF</td></tr> </table> <div style="margin-top: 10px;"> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr><th>b.a</th><th>00</th><th>01</th><th>11</th><th>10</th></tr> <tr><th>d.c</th><td></td><td></td><td></td><td></td></tr> <tr><th>00</th><td>y0</td><td>y1</td><td>y3</td><td>y2</td></tr> <tr><th>01</th><td>y4</td><td>y5</td><td>y7</td><td>y6</td></tr> <tr><th>11</th><td>yC</td><td>yD</td><td>yF</td><td>yE</td></tr> <tr><th>10</th><td>y8</td><td>y9</td><td>yB</td><td>yA</td></tr> </table> <p>4 variables: 16 cases</p> </div>	d c b a	f	0 0 0 0	y0	0 0 0 1	y1	0 0 1 0	y2	0 0 1 1	y3	0 1 0 0	y4	0 1 0 1	y5	0 1 1 0	y6	0 1 1 1	y7	1 0 0 0	y8	1 0 0 1	y9	1 0 1 0	yA	1 0 1 1	yB	1 1 0 0	yC	1 1 0 1	yD	1 1 1 0	yE	1 1 1 1	yF	b.a	00	01	11	10	d.c					00	y0	y1	y3	y2	01	y4	y5	y7	y6	11	yC	yD	yF	yE	10	y8	y9	yB	yA
b a	f																																																																																			
0 0	y0																																																																																			
0 1	y1																																																																																			
1 1	y3																																																																																			
1 0	y2																																																																																			
a	0	1																																																																																		
b	$\overline{a.b}$ y0	$a.b$ y1																																																																																		
1	$\overline{a.b}$ y2	$a.b$ y3																																																																																		
d c b a	f																																																																																			
0 0 0 0	y0																																																																																			
0 0 0 1	y1																																																																																			
0 0 1 0	y2																																																																																			
0 0 1 1	y3																																																																																			
0 1 0 0	y4																																																																																			
0 1 0 1	y5																																																																																			
0 1 1 0	y6																																																																																			
0 1 1 1	y7																																																																																			
1 0 0 0	y8																																																																																			
1 0 0 1	y9																																																																																			
1 0 1 0	yA																																																																																			
1 0 1 1	yB																																																																																			
1 1 0 0	yC																																																																																			
1 1 0 1	yD																																																																																			
1 1 1 0	yE																																																																																			
1 1 1 1	yF																																																																																			
b.a	00	01	11	10																																																																																
d.c																																																																																				
00	y0	y1	y3	y2																																																																																
01	y4	y5	y7	y6																																																																																
11	yC	yD	yF	yE																																																																																
10	y8	y9	yB	yA																																																																																
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr><th>c b a</th><th>f</th></tr> <tr><td>0 0 0</td><td>y0</td></tr> <tr><td>0 0 1</td><td>y1</td></tr> <tr><td>0 1 0</td><td>y2</td></tr> <tr><td>0 1 1</td><td>y3</td></tr> <tr><td>1 0 0</td><td>y4</td></tr> <tr><td>1 0 1</td><td>y5</td></tr> <tr><td>1 1 0</td><td>y6</td></tr> <tr><td>1 1 1</td><td>y7</td></tr> </table> <div style="margin-top: 10px;"> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr><th>b.a</th><th>00</th><th>01</th><th>11</th><th>10</th></tr> <tr><th>c</th><td></td><td></td><td></td><td></td></tr> <tr><th>0</th><td>y0</td><td>y1</td><td>y3</td><td>y2</td></tr> <tr><th>1</th><td>y4</td><td>y5</td><td>y7</td><td>y6</td></tr> </table> <p>3 variables: 8 cases</p> </div>	c b a	f	0 0 0	y0	0 0 1	y1	0 1 0	y2	0 1 1	y3	1 0 0	y4	1 0 1	y5	1 1 0	y6	1 1 1	y7	b.a	00	01	11	10	c					0	y0	y1	y3	y2	1	y4	y5	y7	y6																																														
c b a	f																																																																																			
0 0 0	y0																																																																																			
0 0 1	y1																																																																																			
0 1 0	y2																																																																																			
0 1 1	y3																																																																																			
1 0 0	y4																																																																																			
1 0 1	y5																																																																																			
1 1 0	y6																																																																																			
1 1 1	y7																																																																																			
b.a	00	01	11	10																																																																																
c																																																																																				
0	y0	y1	y3	y2																																																																																
1	y4	y5	y7	y6																																																																																

Appliqué à l'expression $f=b+a.c$, cela donne le résultat ci-contre.

La simplification s'opère de la façon suivante :

- Faire des regroupements de cases les plus grands possibles (intersections premières)
- Réunir ces intersections en ne conservant que ceux qui sont indispensables (éliminer les consensus).

c b a	f
0 0 0	0
0 0 1	1
0 1 0	1
0 1 1	1
1 0 0	0
1 0 1	0
1 1 0	1
1 1 1	1

b.a	00	01	11	10
c				
0	0	1	1	1
1	0	0	1	1

$f=\{1,2,3,6,7\}$
 $f=b + a.\overline{c}$

Le résultat est sous la forme de somme de produit, car nous avons porté notre raisonnement sur les **1**. Si nous travaillons sur les **0**, nous travaillons avec le complément de **f**, par la complémentarité du résultat et l'emploi de De Morgan, le résultat sera sous la forme d'une somme de produit. Le choix d'une des deux méthodes dépend du nombre de **1** et de **0** dans le tableau.

Les regroupement de cases ne peuvent se faire que si l'adjacence algébrique existe, ces cases doivent être en ligne ou en carré.

Exemple : $f(\overline{a}, \overline{b}, c) = \{3, 4, 5, 7\}$

solution par les 1 :

$$f = (3, 7) + (5, 7) + (4, 5)$$

(5, 7) est redondant (consensus)

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

solution par les 0 :

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

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

résolution algébrique :

$$f(a, b, c) = \{3, 4, 5, 7\} = a.b.\overline{c} + \overline{a}.\overline{b}.c + a.\overline{b}.c + a.b.c$$

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

L'expression est biforme en a, le groupe (5, 7) est bien le consensus de b.

	b.a			
c	00	01	11	10
0	0	0	1	0
1	1	1	1	0

Exercice : soit $f(a, b, c, d) = \{2, 6, E\}$ défini par les 1 et $f = \{5, D\}$ défini par les 0, le reste n'est pas déterminé.

Représentez le tableau de Karnaugh, résolvez par les 1 et par les 0.

d	c	b	a	f
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

	b.a			
d.c	00	01	11	10
00				
01				
11				
10				

II - Méthodes algébriques.

II-1) Emploi des théorèmes de l'algèbre de Boole :

Le tableau ci-dessous résume les principaux théorèmes de l'algèbre de Boole que l'on utilisera cette année.

En résumé et à retenir :

	a) ET	b) OU
1) Élément neutre	$a.1=a$	$a+0=a$
2) Élément absorbant	$a.0=0$	$a+1=1$
3) Idempotence	$a.a=a$	$a+a=a$
4) Complément	$a./a=0$	$a+/a=1$
5) Commutativité	$a.b=b.a$	$a+b=b+a$
6) Associativité	$a.(b.c)=(a.b).c=a.b.c$	$a+(b+c)=(a+b)+c=a+b+c$
7) Distributivité	$a.(b+c)=a.b+a.c$	$a+(b.c)=(a+b).(a+c)$

8) Relations diverses	$a.(a+b)=a$ $a./(a+b)=a.b$ $a./(a+b)=0$ $a./(a.b)=a./b$	$a+(a.b)=a$ $a+/(a.b)=a+b$ $a+/(a.b)=1$ $a+/(a+b)=a+/b$
9) De Morgan	$/(a.b)=/a+/b$ $a.b=/(/a+/b)$	$/(a+b)=/a./b$ $a+b=/(/a./b)$
10) Fonction biforme	$a.b+/a.c=(a+c).(/a+b)$	$(a+b).(/a+c)=a.c+/a.b$
11) Consensus	$a.b+/a.c+b.c=a.b+/a.c$	$(a+b).(/a+c).(b+c)=(a+b).(/a+c)$
12) Consensus généralisé	$a.b+/a.c+b.c.d=a.b+/a.c$	$(a+b).(/a+c).(b+c+d)=(a+b).(/a+c)$

dualité

Il est important de remarquer la dualité entre le OU et le ET, en transposant les ET avec les OU et les 1 avec les 0.

Exercice : vérifiez chaque relation sur un Tableau de Karnaugh ou algébriquement.

Distributivité, éliminer les redondances, emploi judicieux de De Morgan, consensus, fonction biforme carrée, adjonction d'éléments neutres et mise en facteur, tenir compte éventuellement de la nature des portes servant à la réalisation, ..., voir TD.

II-2) Méthode du consensus

Il faut prendre tour à tour chaque variable, rechercher ses consensus s'ils existent, nous pouvons supprimer les termes contenant ces consensus. Le résultat final est l'apparition d'intersections premières, ...voir TD.

III - Exercices.

1°) Vérifiez avec les tableaux de Karnaugh les théorèmes de distributivité.

2°) Simplifiez :

$$y1 = \bar{a}.b.c + a.c + (a+b).\bar{c}$$

$$y2 = b.c + a.c + a.b + b$$

$$y3 = (a./b+c)(a+/b).c$$

$$y4 = (a.c+b./c).(a+/c).b$$

$$y5 = (!a\&b\#a\&!b)\&(a\&b\#!a\&!b)$$

3°) Complémentez puis simplifiez :

$$A = a.b + b.c + a.c$$

$$B = \bar{c}.\bar{d} + \bar{a}.\bar{b} + c.\bar{d} + a.\bar{b}$$

HORS TD

$$y6 = a.b.c + a.b.\bar{c} + \bar{a}.b.\bar{c} + a.\bar{b}.c$$

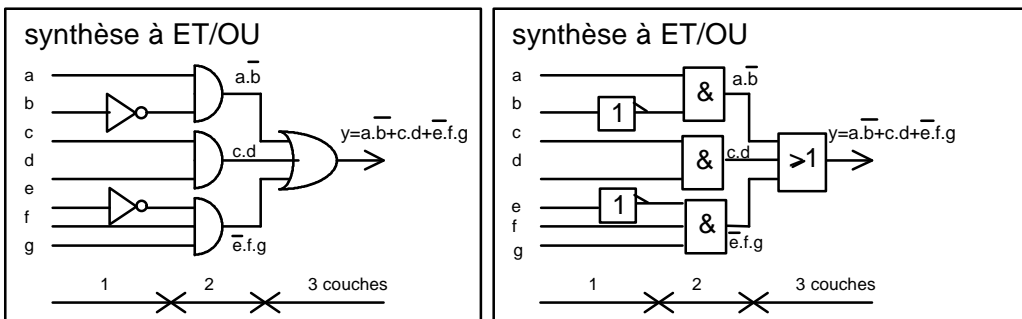
$$y7 = a./b./c + a.b./c + a.b.c + a.b./c$$

TD4 - Réalisation des circuits.

Selon les technologies électroniques (TTL, CMOS, ECL,...), nous rencontrerons plus fréquemment des ET-Non ou des OU-Non, car une version peut être plus facile à réaliser, ou bien elle peut être plus rapide. D'autre part, les montages sont souvent très complexes et doivent réaliser de grande quantité d'opérations en une seconde ; il est donc nécessaire d'économiser le nombre des fonctions et d'améliorer si possible la rapidité. Selon les technologies la rapidité de résolution d'une porte varie de quelques 100 ns à quelques 1ns, la mise en pratique des technologies rapides est difficile. Notre objectif est de résoudre de façon optimale les schémas.

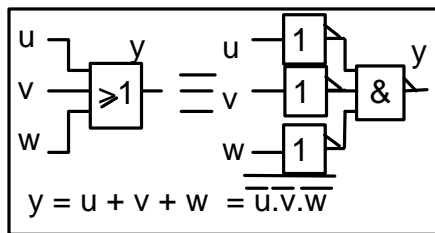
I - Synthèse avec la structure ET/OU.

Après simplification algébrique ou par Karnaugh, la fonction doit être mise sous forme de somme de produit (pas nécessairement canonique). Les schémas ainsi obtenus possèdent au maximum trois couches de circuits. Cela est important pour minimaliser le temps de propagation.

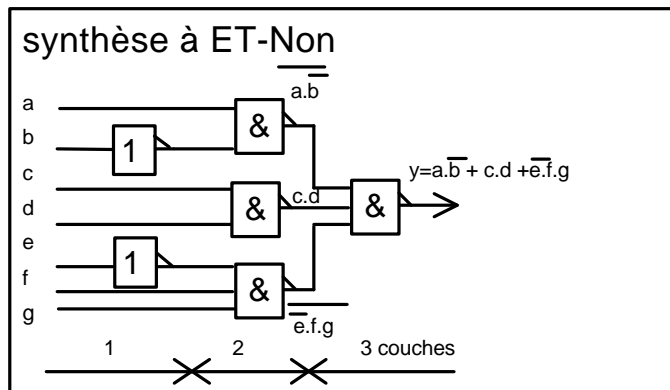


II - Synthèse à ET-Non.

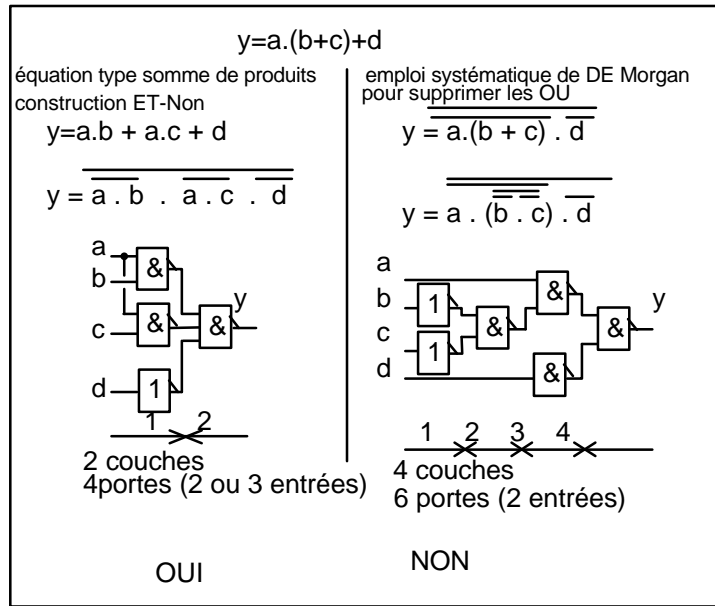
La structure précédente amène naturellement à celle à ET-Non. Le théorème de De Morgan permet cette transformation. Le montage reste constitué de trois couches.



De Morgan

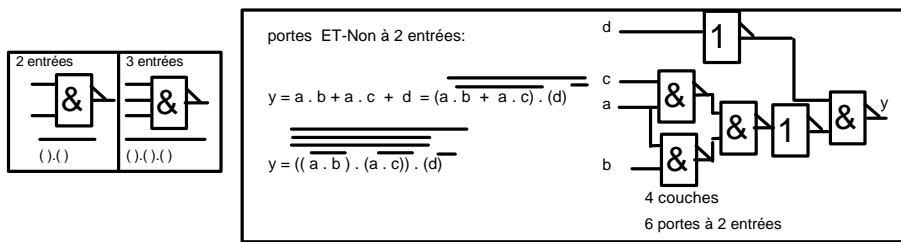


REMARQUE : Il est important de passer par une forme " somme de produit " simplifiée, pas nécessairement canonique (car pas simplifiée). Il ne faut pas partir d'une expression quelconque et chercher à supprimer les OU par un emploi abusif de De Morgan.



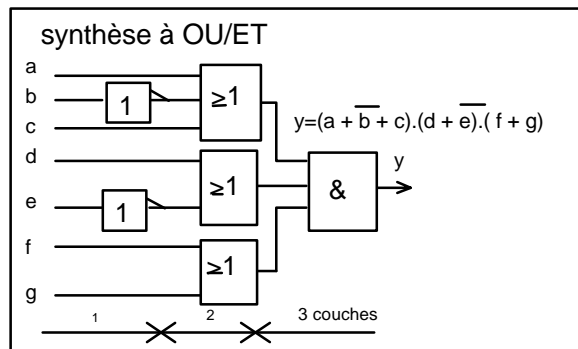
III - Portes ET-Non limitées par le nombre d'entrée.

Lorsqu'on choisit un circuit intégré, il comporte plusieurs portes du même type (ex. en TTL : le 7400 possède 4 portes à deux entrées). Parfois, il n'est plus possible de réaliser la synthèse en trois couches, on travaillera alors sur des groupes de termes égaux au nombre d'entrée des portes choisies.



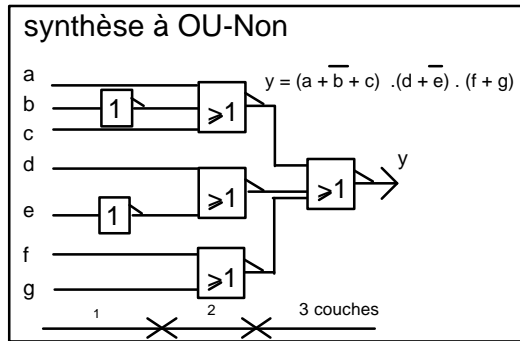
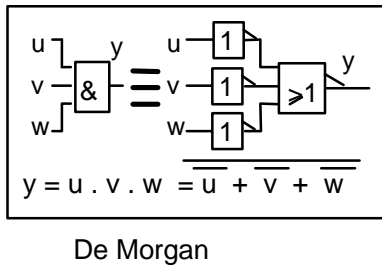
IV - Synthèse avec la structure OU/ET.

C'est la méthode duale de celle à ET/OU. Après simplification algébrique ou par Karnaugh, la fonction doit être mise sous forme de produit de somme (pas nécessairement canonique). Les schémas ainsi obtenus possèdent au maximum trois couches de circuits. Cela est important pour minimaliser le temps de propagation.

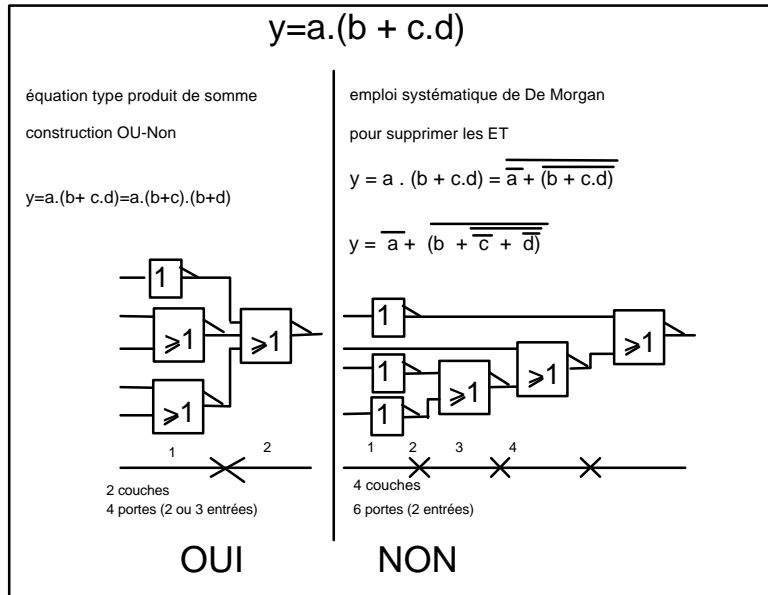


V - Synthèse à OU-Non.

La structure précédente amène naturellement à celle à OU-Non. Le théorème de De Morgan permet cette transformation. Le montage reste constitué de trois couches.

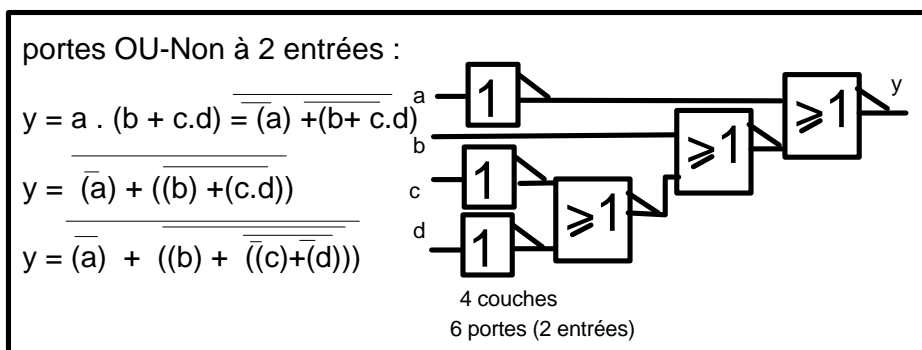


REMARQUE : Il est important de passer par une forme "produit de somme" simplifiée, pas nécessairement canonique (car pas simplifiée). Il ne faut pas partir d'une expression quelconque et chercher à supprimer les ET par un emploi abusif de De Morgan.



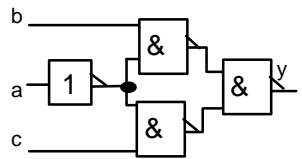
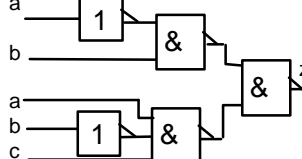
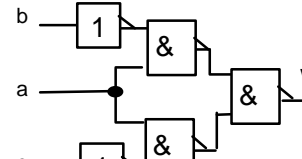
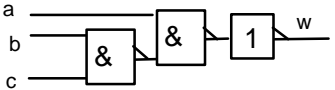
VI - Portes OU-Non limitées par le nombre d'entrée

Lorsqu'on choisit un circuit intégré, il comporte plusieurs portes du même type (ex. en TTL :le 7402 possède 4 portes à deux entrées). Parfois, il n'est plus possible de réaliser la synthèse en trois couches, on travaillera alors sur des groupes de termes égaux au nombre d'entrée des portes choisies.

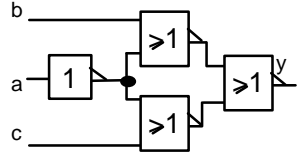
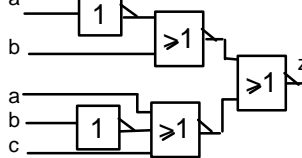
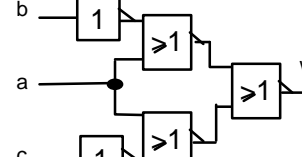
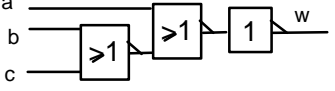


VII - Optimisations

Optimisation des fonctions à ET-Non

 <p>mettre en commun des termes</p>	 <p>mettre en commun des ET-Non en utilisant $\overline{a.b} = \overline{a}. \overline{b}$ ex : $z = \overline{a.b} + a.b.c = \overline{a}. \overline{b} + a . a.b . c$ (théorème 8-4a)</p>
	<p>faire apparaître un ET-Non</p> $w = a . \overline{b} + a . \overline{c}$ $= a . (\overline{b} + \overline{c})$ $= a . \overline{b.c}$ 

Optimisation des fonctions à OU-Non

 <p>mettre en commun des termes</p>	 <p>mettre en commun des ET-Non en utilisant $\overline{a+b} = \overline{a} . \overline{b}$ ex : $z = \overline{(a+b)}.(\overline{a+b+c}) = (\overline{a+b} + b).(\overline{a} + \overline{a+b} + c)$ (théorème 8-4b)</p>
	<p>faire apparaître un ET-Non</p> $w = (a + b).(\overline{a} + \overline{c})$ $= a + (\overline{b} . \overline{c})$ $= a + \overline{b.c}$ 

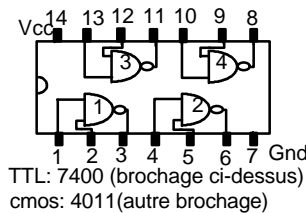
VIII - Exercices

1°) Elaborez les schémas à ET-Non puis à OU-Non en 3 couches maxi. et optimum en nb. de portes :

	ET-Non	OU-Non
1: $y = a.(b+c)$	3	3
2: $y = a.b + /c$	2	4
3: $y = /a.b + a.c$	4	4
4: $y = /a.b + a./b$	4	5
5: $y = ((/a+b).(a+c) + b.c).(a+b+c)$	4	4
6: $y = /a.b./c + a./b.c$	5	7
7: $y = !a\&(c\# b\&d)$	4	3
8: $y = (!a\#\!b)\&(c\#d)$	4	5
9: $y = /a.b + a./b.c$	4	6
10: $y = a.(/b+c) + /b.c$	4	5
11: $y = /A.E0 + A.E1$ (multiplexeur)	4	4
12: $y = a./b + a.c + b./c$	4	4
13: $y = /a.b + a./b + a.c$	5	5
14: $y = a./c + b./c$	4	2
15: $y = (/a+b).(a+/b)$	5	4

2°) Réalisez avec des ET-Non à 2 entrées seulement:

- 1: $y = a.b + b.c + c.d$
- 2: $y = a.b.c$
- 3: $y = a+b+c$



TD 5 - Méthode du SI-ALORS.

Nous avons appris jusqu'à présent à simplifier, et à réaliser de manière matérielle les fonctions logiques. Malheureusement, dans les problèmes concrets, les fonctions logiques ne sont pas souvent données directement par des équations logiques. Nous allons donc maintenant examiner comment passer d'un cahier des charges à des fonctions logiques.

I - Présentation du tableau SI-ALORS

I-1) Table de vérité

Nous avons déjà eu l'occasion de parler des tables de vérité. Ici, nous allons nous contenter de remarquer qu'une table de vérité peut être vue sous un autre angle.

Par exemple écrire :

a	b	s
0	0	0
0	1	1
....

peut s'interpréter :

SI a=0 et b=0 ALORS s=0

SI a=0 et b=1 ALORS s=1.....

Nous concluons :

- une table de vérité est équivalente à une équation booléenne,
- une table de vérité est équivalente à une liste de conditions SI ... ALORS ...

Nous nous demandons :

Si nous appelons une table de vérité un tableau dans lequel ne figurent que des uns et zéros, est-il possible d'inventer un tableau dans lequel figurent des uns, des zéros et des variables ?

I-2) Tableau SI-ALORS

Un tableau SI-ALORS sera un tableau composé de deux parties : une partie condition (SI) et une partie conclusion (ALORS). La partie conditions pourra être composée par des équations booléennes quelconques ayant comme variables les entrées (et les entrées seulement), mais pour simplifier nous prendrons par la suite cette partie comme celle d'une table de vérité c'est à dire ne comportant que des uns et zéros. La partie conclusion sera composée d'équations booléennes simples (du genre $s=1$) ou sorties en fonction des entrées. D'autre part lorsqu'aucune spécification n'est faite pour certaines entrées, il sera supposé que la sortie est alors nulle, sauf dans le cas où la sortie est spécifiée à zéro.

Exemple :

Soit un circuit comprenant comme entrée e1 et e2 et comme sortie s.

Une équation SI pourrait être $e1=e2$, une équation ALORS pourrait être $s=1$.

(On suppose naturellement que si la condition n'est pas vérifiée la sortie sera $s=0$)

SI		ALORS
e1	e2	s
1	1	1

D'un tableau SI-ALORS, il est toujours possible de trouver une équation booléenne.

Donner ici une méthode systématique pour passer d'un tableau SI-ALORS à une équation booléenne serait trop long et fastidieux. Nous allons plutôt donner les exemples les plus courants que l'on rencontrera par la suite.

II - Exemples

Les entrées seront notées e1, e2, ... ei et la sortie s.

Les exemples généraliseront la table de vérité par ordre de difficulté croissante.

SI		ALORS
e1	e2	s
1	1	1

(les =1 ne sont pas écrits par simplification) Equation logique (E.L.)

SI e1.e2 alors s donne $s=e1.e2$

(Si la condition n'est pas vérifiée $s=0$)

Dans cet exemple on a tout simplement une table de vérité. Abordons les problèmes plus difficiles:

SI	ALORS
e3 e2 e1	s
0 1 1	e4+/e5
1 1 1	/e4+e5
1 0 1	/e5

La difficulté est qu'ici il faut simplifier algébriquement.

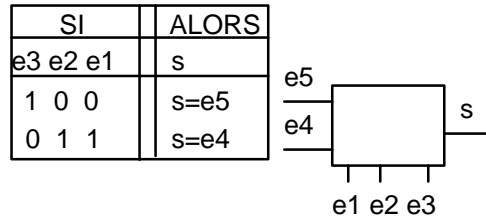
E.L. : $s = (e4+/e5)(e1.e2./e3) + (/e4+e5)(e1.e2.e3) + /e5(e1./e2.e3)$

Doit éventuellement se simplifier.

La grande nouveauté est l'apparition d'équations logiques dans la partie ALORS.

Remarques IMPORTANTES :

- un tableau SI-ALORS est parfois utilisé pour décrire des circuits un peu compliqués. Il est quand même appelé table de vérité (truth table) par les fabricants de composants. Tout circuit un peu compliqué (plus de 5 entrées) ne peut qu'être décrit par ce genre de tableau.
- Nous utiliserons par la suite la convention montrée ci-contre pour tous les tableaux SI-ALORS : les conditions SI porteront toujours sur des entrées "spéciales" dessinées de bas en haut. Elles sont parfois appelées entrées de programmation.



III - Tables SI-ALORS et le langage ABEL

Il est possible de spécifier des tables SI-ALORS avec Abel mais on ne peut pas le faire à l'intérieur d'une table de vérité. Il faut spécifier cela à l'aide des équations. Par exemple :

Vous pouvez constater ci-contre la facilité avec laquelle on spécifie une table SI-ALORS à l'aide de la structure when ...then...else. Notez que le test de l'égalité se fait comme en langage C par l'opérateur ==.

<pre> MODULE EXO6 title 'exercice 6 : MUX 2 vers 1' e1,e2,A pin;"entrees out pin istype 'com'; "sorties X=X.; equations when !A then out=e1; else out=e2; test_vectors ([A, e1, e2] ->out) [0, 0, 0] -> X; " ajoutez vos vecteurs ici [1, 1, 1] -> X; end </pre>	<pre> MODULE EXO7 title 'exercice 7 : MUX 4 vers 1 par la methode du SI ALORS' e0,e1,e2,e3,A,B pin;"entrees out pin istype 'com'; "sorties selecteur = [B, A]; equations when selecteur==0 then out=e0; when selecteur==1 then out=e1; when selecteur==2 then out=e2; when selecteur==3 then out=e3; end </pre>
--	---

Le langage Abel permet de spécifier des équations grâce à des opérateurs relationnels permettant ainsi en une seule ligne de spécifier beaucoup de données. Nous vous donnons ci-contre des spécifications que l'on vous demande de comprendre. Ce n'est pas à proprement parler la méthode du SI-ALORS présentée plus haut mais ce genre de spécifications est très utile.

```

module M6809A
title '6809 memory decode Jean Designer Data I/O Corp Redmond WA'
  A15,A14,A13,A12,A11,A10 pin 1,2,3,4,5,6;
  ROM1,IO,ROM2,DRAM pin 14,15,16,17;
  H,L,X = 1,0,.X.;
  Address = [A15,A14,A13,A12, A11,A10,X,X, X,X,X,X, X,X,X,X];
equations
  !DRAM = (Address <= ^hDFFF);
  !IO = (Address >= ^hE000) & (Address <= ^hE7FF);
  !ROM2 = (Address >= ^hF000) & (Address <= ^hF7FF);
  !ROM1 = (Address >= ^hF800);
test_vectors
  (Address -> [ROM1,ROM2,IO,DRAM])
  ^h0000 -> [ H, H, H, L ];
  ^h4000 -> [ H, H, H, L ];
  ^h8000 -> [ H, H, H, L ];
  ^hC000 -> [ H, H, H, L ];
  ^hE000 -> [ H, H, L, H ];
  ^hE800 -> [ H, H, H, H ];
  ^hF000 -> [ H, L, H, H ];
  ^hF800 -> [ L, H, H, H ];
end M6809A
    
```

IV - Exercices.

Nous allons montrer d'abord avec les trois premiers exercices (**un seul sera traité**), que pour certains problèmes pratiques l'utilisation de la méthode SI-ALORS se réduit à l'utilisation d'une table de vérité.

1°) Une société est composée de 4 actionnaires ayant les nombres suivants d'actions : A=60 , B=100, C=160, D=180.

Nous désirons construire une machine à voter automatiquement, tenant compte dans le résultat du poids en actions de chaque personne. La machine dispose de quatre boutons poussoirs (A, B, C, D). Le résultat sera un voyant (V) qui s'allumera si la majorité pondérée appuie sur les boutons.

Après une étude de $V = f(A,B,C,D)$, construire une machine avec des ET-Non.

2°) Une automobile dispose de 4 commandes de feux : v veilleuses, c croisement, r route, a antibrouillards, qui prennent l'état 1 si on les actionne. Les phares concernés (V,C,R,A) sont à l'état 1 (allumés) selon la logique suivante :

- les feux V, C, R, A ne peuvent être allumés ensembles,
- les feux de croisement sont prioritaires sur ceux de route et d'antibrouillard,
- les antibrouillards sont prioritaires sur ceux de route,
- les veilleuses peuvent être allumées seules, mais l'allumage des autres entraîne celui des veilleuses.

Donnez les tables de vérité et DK de V,C,R,A et les schémas à ET-Non, puis à OU-Non, puis à ET-Non à 2 entrées seulement.

3°) Dans une usine de briques, la qualité de celles-ci se fait selon quatre critères :

Poids P, longueur L, largeur l, hauteur H, (0 : pas correct ,1: correct)

Les briques sont classées en quatre catégories:

*** A : le poids et 2 dimensions au moins sont corrects.

** B : seul le poids est mauvais OU le poids est correct mais au moins deux dimensions sont fausses.

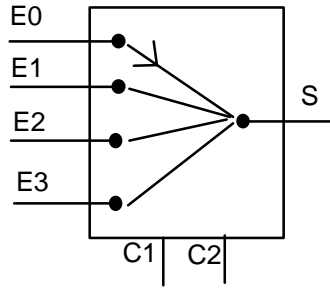
* C : le poids est incorrect et une dimension seulement est incorrecte.

D: refus: les autres cas.

Faites l'étude des DK et des schémas à ET-Non.

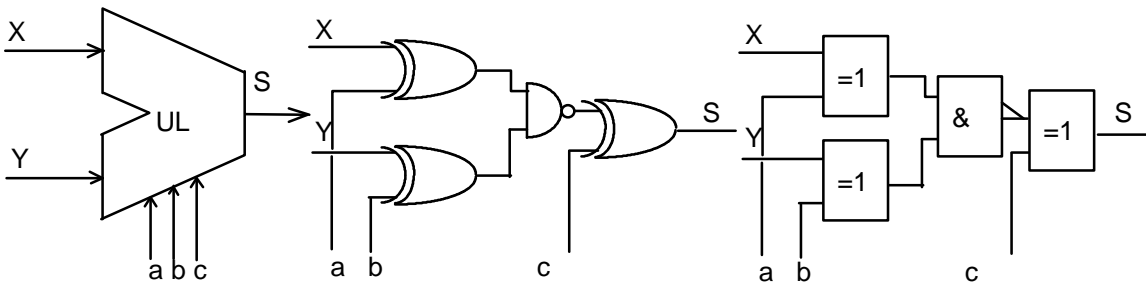
Le quatrième exercice et les suivants montrent, par contre, que dans certains cas la méthode du SI-ALORS est beaucoup plus efficace.

4) Un multiplexeur (composant sur lequel nous reviendrons plus tard) réalise une fonction complexe pouvant être schématisée ci-dessous :



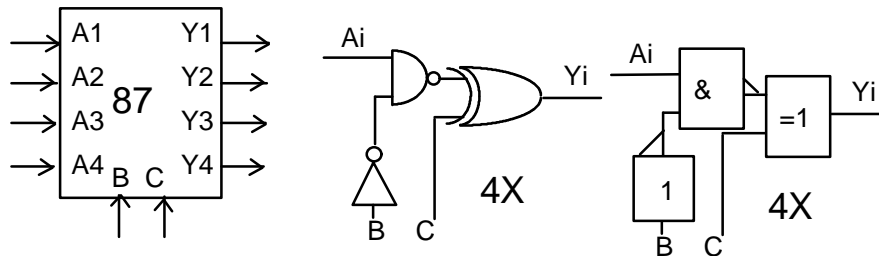
- Combien de lignes comporterait une table de vérité décrivant ce composant ?
- Montrer qu'un tableau SI-ALORS de 4 lignes peut décrire ce composant.
- En déduire l'équation logique de ce multiplexeur.

5) Unité logique : il s'agit d'introduire la notion de fonction programmable. Etablir la nature des huit fonctions $F(A,B)$ programmables selon a, b, c .



Montrer qu'une présentation en SI-ALORS convient bien à ce genre de problème.
Ecrire le programme ABEL correspondant.

6) Le circuit intégré TTL 7487 est un générateur de fonctions. Quelle est la nature des fonctions programmables ?



Mettre les résultats dans un tableau SI-ALORS.

NOM :
Groupe :
 Novembre 1995

Devoir surveillé n°1 d'informatique industrielle (extraits)

Durée 2 heures
FEUILLE REPONSE n°1

Exercice 1.

On donne l'expressions :

$$S1 = (A.B./C.D + C./D)/(B./A + D)$$

1°) Remplir le tableau de Karnaugh ci-dessous

	BA	00	01	11	10
DC					
00					
01					
11					
10					

S1

3°) Donner la forme conjonctive canonique

Réponse : S1 =

4°) Donner la forme conjonctive simplifiée

Réponse : S1 =

5°) Ecrire la fonction S1 sous forme numérique par les 1 avec A poids faible et D poids fort

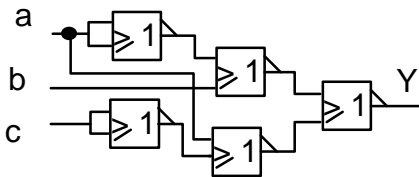
Réponse : S1 =

2°) Donner la forme simplifiée disjonctive.

Réponse : S1 =

Exercice 2.

On donne le schéma en OU-NON suivant :



3°) Ecrire la fonction Y sous forme disjonctive canonique en fonction de a,b,c

Réponse : Y =

4°) Ecrire la fonction Y sous forme disjonctive simplifiée.

Réponse : Y =

1°) Ecrire Y en fonction de a, b et c :

Réponse : Y =

5°) Réaliser un schéma en NANDS (3 couches maxi) correspondant à la fonction Y

Réponse : schéma à réaliser ci-dessous

2°) Remplir le tableau de Karnaugh ci-dessous :

	ba	00	01	11	10
c					
0					
1					

Y

Exercice 3.

Soit la fonction booléenne de 5 variables : $Z = \bar{a}.b.\bar{c} + a.\bar{d} + b.\bar{c}.\bar{d}.e + a.\bar{d}.e$

1°) Simplifier de manière algébrique cette fonction booléenne.

Réponse :

2°) Passer ensuite la fonction Z sous forme conjonctive à l'aide de la distributivité de l'addition par rapport à la multiplication.

Réponse :

NOM :

Prénom :
Groupe :

FEUILLE DE REPONSE N°2

Exercice 4 (cours).

On rappelle que le passage de la base 10 avec une précision fractionnaire de 10^{-x} vers la base 2 avec une précision fractionnaire de 2^{-n} au moins aussi bonne se fait avec $n > 3,32.x$

1°) Quelle est la précision fractionnaire de 37,8

Détail de calcul de 37,8 en base 2 :

Réponse :

2°) calculer ce nombre en base 2.

Réponse :

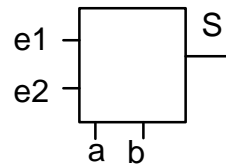
3°) Représenter -1 en complément à deux sur 4 bits.

Réponse :

--	--	--	--

Exercice 5. (Méthode du SI-ALORS)

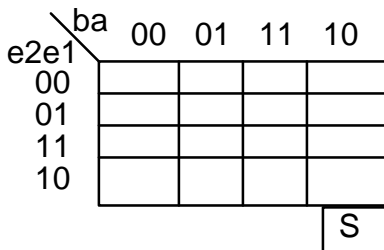
On veut réaliser une unité logique programmable dont le schéma fonctionnel est donné ci-contre. Son fonctionnement est décrit par la table SI-ALORS détaillée ci-dessous



SI		ALORS
a	b	S =
0	0	$e1.e2$
0	1	$/(e1.e2)$
1	0	$e1+e2$
1	1	$/(e1+e2)$

1°) Donner l'équation de $S=f(a,b,e1,e2)$ sous forme disjonctive.

Réponse :



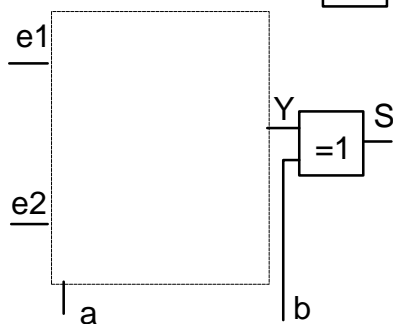
2°) remplir le tableau de Karnaugh ci-contre correspondant à $S=f(a,b,e1,e2)$.

3°) En déduire une équation simplifiée sous forme disjonctive.

Réponse :

4°) Combien de portes ET-NON faut-il pour un schéma trois couches ?

Réponse :



5°) On préfère réaliser le circuit à l'aide d'un OU-EXCLUSIF et de portes ET-NON comme indiqué ci-contre. En vous aidant de la table SI-ALORS d'un OU-EXCLUSIF utilisée en TP et TD, écrire l'équation simplifiée $Y=f(a,e1,e2)$ et compléter le schéma ci-contre à l'aide d'un schéma en ET-NON.

Réponse : $Y =$

Cours 5 - Circuits de transcodage.

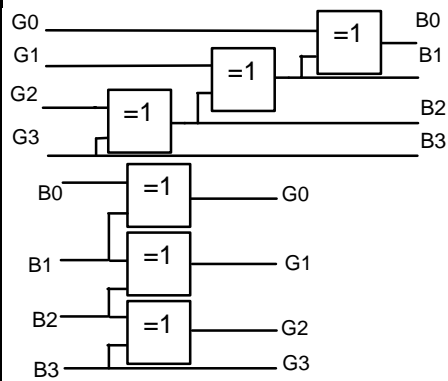
Le codage désigne l'ensemble des codeurs (2^n entrées, n sorties), des décodeurs (n entrées, 2^n sorties) et des transcodeurs (p entrées et k sorties, k et p quelconques). Ils transforment une information présente à leurs entrées sous une forme donnée (code 1) en une information présente à leurs sorties sous une autre forme (code 2).

I - Réalisation d'un circuit de transcodage.

Quand le circuit n'existe pas tout fait, on le calcule et on le réalise selon les méthodes apprises précédemment.

Exemple : Transcodeur binaire/Gray Gray/binaire.

n	Code binaire				Code gray			
	B3	B2	B1	B0	G3	G2	G1	G0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0



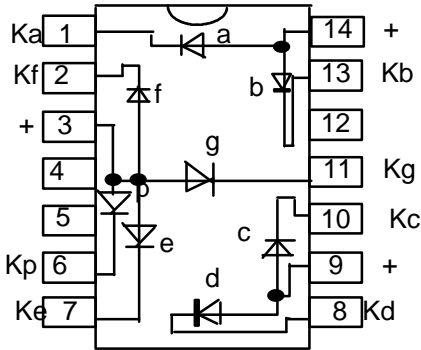
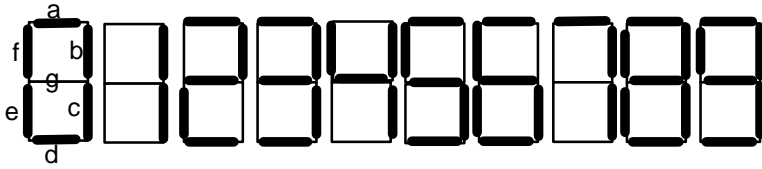
II - Circuits existants.

On trouve dans le commerce quelques types de circuits dont l'emploi en grand nombre justifie une fabrication en série. Nous allons en présenter quelques uns. On trouvera tous les autres dans les catalogues de feuilles de données des fabricants.

- Exemples** :
- Décodeurs : Binaire/décimal -->74154
 BCD/Décimal -->7442
 Exces 3/Décimal -->7443
 Exces 3 Gray/Décimal -->7444
 - Transcodeurs : BCD/Binaire -->74184
 Binaire/BCD -->74185
 BCD/7 segments -->7447
 - Codeurs : Décimal/Binaire -->74148

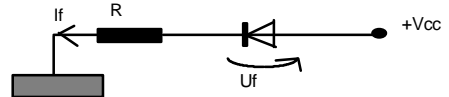
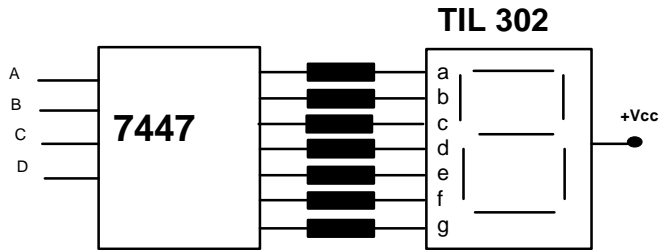
Exemple : décodeurs d'affichage 7447.

L'afficheur lumineux à 7 segments (TIL 302) est un circuit intégré formé de 7 diodes lumineuses en forme de bâtonnets permettant de représenter tout chiffre de 0 à 9.



Entrées	fonctions
1	cathode a
2	cathode f
3	+ 5 V
6	cathode du pt.
7	cathode e
8	cathode d
9	+ 5 V
10	cathode c
11	cathode g
13	cathode b
14	+ 5 V

L'allumage d'un segment se fait donc par une mise à zéro de la cathode (K) qui joue le rôle d'entrée, l'anode (+) étant à + 5V.



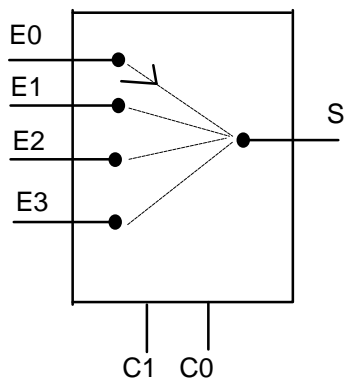
$V_f \approx 1,2 \text{ V}$ $V_{cc} = 5 \text{ V}$
 $I_f = 10 \text{ à } 200 \text{ mA}$
 $R = (V_{cc} - V_f) / I_f = 330 \Omega$

Il existe d'autres circuits décodeurs 7 segments 4511 et 4543 en CMOS.

Cours 6 - Multiplexage - Demultiplexage.

I - Définitions.

Un multiplexeur est un circuit réalisant un aiguillage de l'une des entrées vers une sortie unique.



C1	C0	S
0	0	E0
0	1	E1
1	0	E2
1	1	E3

SI	ALORS
C1=0.C0=0	S=E0
C1=0.C0=1	S=E1
C1=1.C0=0	S=E2
C1=1.C0=1	S=E3

Deux représentations possibles du fonctionnement

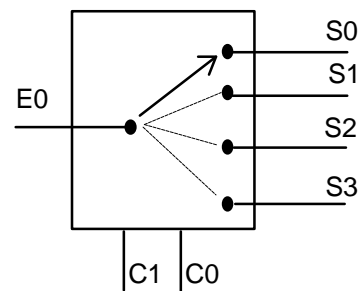
La sortie s'exprime directement en fonction des entrées

$$S = E0C1C0 + E1C1\bar{C0} + E2C1\bar{C0} + E3C1C0$$

La position de l'interrupteur est fixée par une commande

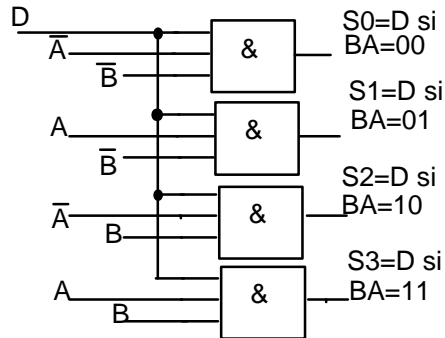
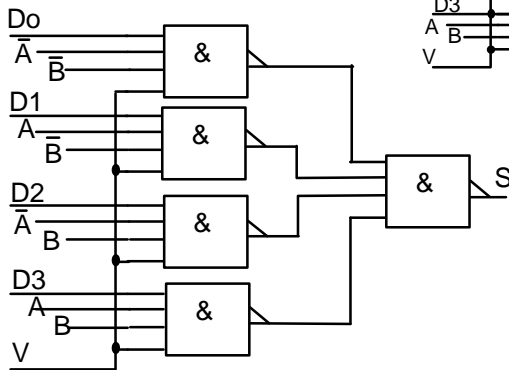
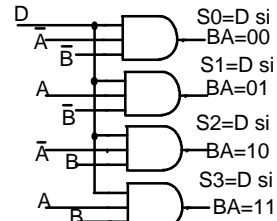
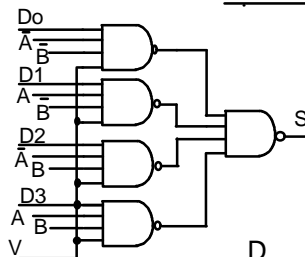
(2ⁿ entrées => n éléments binaires de commande)
On a une entrée validation V : V=1 multiplexage, V=0 S=0

Un démultiplexeur est un circuit qui distribue l'information d'entrée vers l'une des sorties sélectionnée.

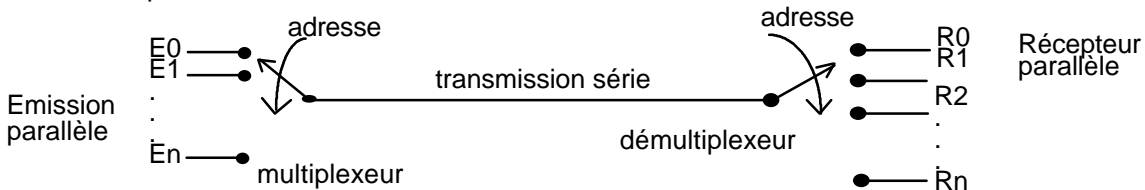


Si l'entrée est toujours égale à 1, le démultiplexeur fonctionne comme un décodeur binaire.

Schémas



Utilisation : si l'on envoie à distance les informations issues d'un grand nombre de sources différentes, on multiplexe ces informations pour les transmettre en série sur une seule ligne. A l'autre extrémité, il faut démultiplexer.



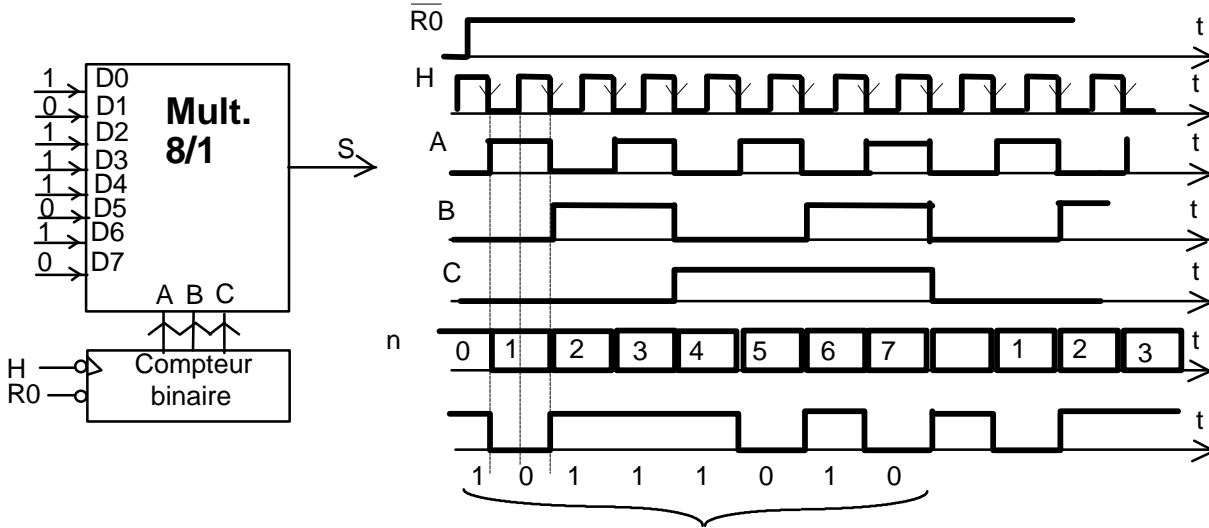
Avec la structure précédente, on trouve commercialisés :

- multiplexeur simple à 8 (ou 16 entrées) -> 74151 - (74150)
- double multiplexeur 2x4 entrées et 2 sorties -> 74153

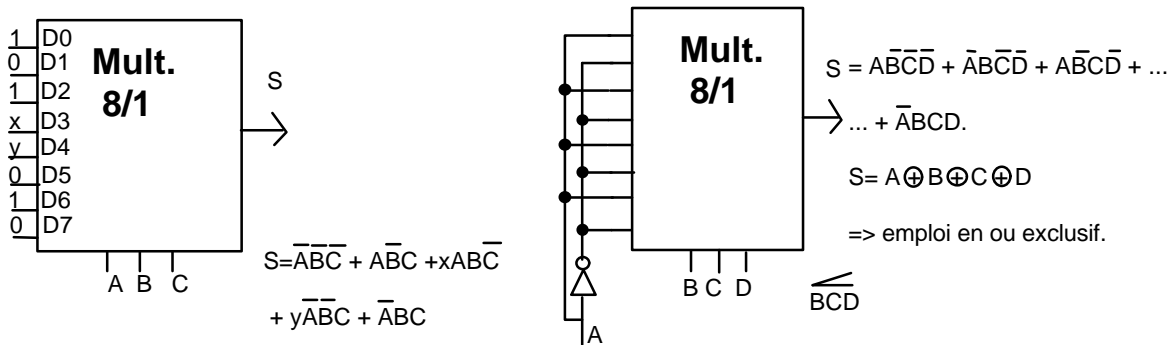
- quadruple multiplexeur à 4x2 entrées et 4 sorties -> 74157
- Pour les démultiplexeurs, on emploie des décodeurs avec entrée sur la validation.
- décodeur binaire/décimal -> 74139 à 1 entrée et 4 sorties,
- > 74138 à 1 entrée et 8 sorties

II - Application particulières.

*** Transformation parallèle/série :**



*** Générateur de fonction logique combinatoire (1ere forme canonique).**



Cours 7 - Paramètres électriques et temporels.

Le but de ce cours est de dégager l'utilité pratique des paramètres électriques et temporels pour la réalisation de systèmes. Nous passons en revue pour cela, les familles logiques, les potentiels d'entrée et de sortie et les temps de retard.

I - Familles et sous familles.

Il existe aujourd'hui essentiellement quatre familles logiques : TTL (Transistor Transistor Logic), ECL (Emitter Coupled Logic), CMOS (Complementary Metal Oxide Semi Conductor), GaAs (Arséniure de Gallium). Elles se caractérisent essentiellement par leur vitesses de fonctionnement : à une vitesse élevée correspond généralement une consommation élevée.

I-1) TTL.

Famille la plus développée, fonctionne en mode bloqué-saturé.

TTL L (Low Power)

TTL LS (Low Power Schottky)

TTL ALS (Advanced Low Power Schottky) : boîtier 4 NAND consomme 1,9 mA au maximum sous 5 V, et une porte commute en 11 ns.

I-2) ECL

Fonctionne en régime linéaire.

ECL 10KH : boîtier 4 NOR consomme 23 mA sous -5,2V les portes ayant un retard de 1,55 ns au maximum.

I-3) CMOS

Emploie des transistors à effet de champ.

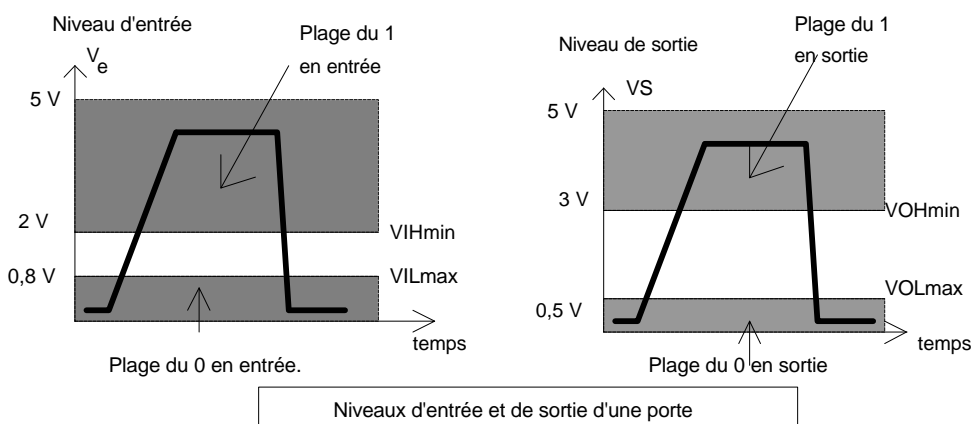
Famille HC fonctionne pour une plage de tension 2 à 6 V : boîtier 4 NAND consomme 20 microA au repos sous 4,5 V et les portes commutent en 25 ns. La consommation croit avec le fréquence d'utilisation. Le temps de retard diminue si la tension d'alimentation croit : 125 ns pour 2 V et 21 ns pour 6 V. Famille HCT fonctionne sous 5V (compatible TTL)

II - Paramètres électriques.

II-1) Niveaux d'entrée et de sortie.

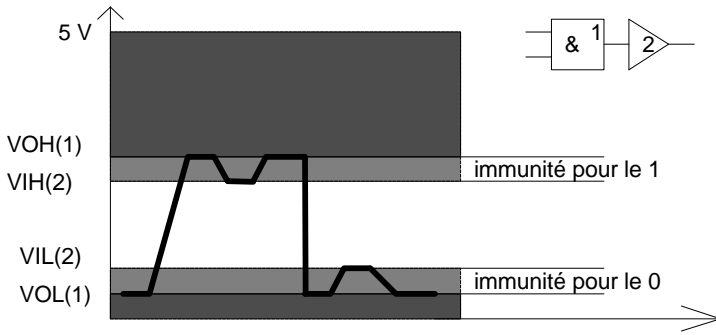
Les sigles suivants définissent les niveaux d'entrée et de sortie :

- V_{IH} (High Level Input Voltage) : tension d'entrée au niveau haut,
- V_{IL} (Low Level Input Voltage) : tension d'entrée au niveau bas,
- V_{OH} (High Level Output Voltage) : tension de sortie au niveau haut,
- V_{OL} (Low Level Output Voltage) : tension de sortie au niveau bas.



Comme la sortie d'une porte alimente l'entrée d'une autre porte, il faut que la plage de reconnaissance en entrée soit plus large que la plage de variation du signal de sortie de façon à disposer d'une marge de sécurité.

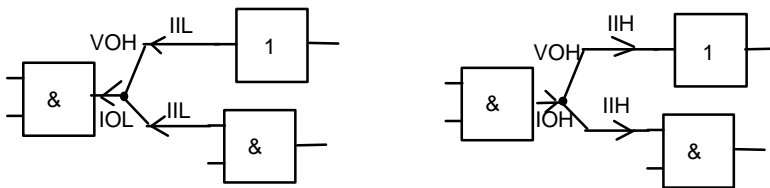
II-2) Immunité aux bruits



II-3) Courants de sortie et d'entrée.

Les sigles suivants désignent les courants d'entrée et de sortie.

- I_{IH} (High Level Input Current) : courant d'entrée au niveau haut,
- I_{IL} (Low Level Input Current) : courant d'entrée au niveau bas,
- I_{OH} (High Level Output Current) : courant de sortie au niveau haut,
- I_{OL} (Low Level Output Current) : courant de sortie au niveau bas.

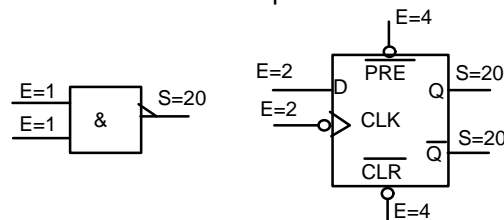


Sens des courants absorbés et fournis par des portes TTL

L'entrance (fan-in) vaut 1 pour une NAND.

La sortance (fan-out) est le nombre maximal d'entrées qu'une sortie peut alimenter : c'est le plus petit des rapports I_{OH}/I_{IH} ou I_{OL}/I_{IL}

L'assemblage des circuits suit une règle très simple : il suffit que la sortance d'un circuit soit supérieur ou égale à la somme des entrances des circuits qu'il commande.



Entrances et sortances de circuits.

II-4) Courant de court-circuit.

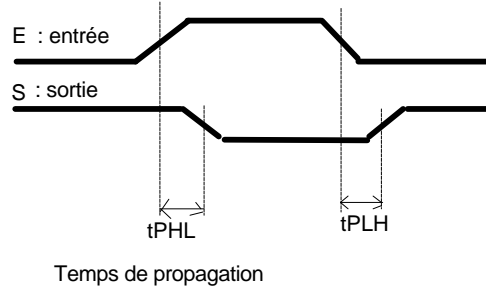
Il est noté I_{OS} (Short Circuit Output Current) est fourni par une sortie, normalement à l'état 1, mais forcée à l'état 0 par un court-circuit à la masse (peut être destructif).

Pour une TTL ALS il vaut 140 mA (contre 400 micro normalement).

III - Paramètres temporels.

Deux paramètres caractérisent le retard d'un circuit :

- t_{PHL} (propagation delay time, high to low level), temps de propagation pour la transition descendante du signal de sortie,
- t_{PLH} (propagation delay time, low to high level), temps de propagation pour la transition ascendante du signal de sortie.

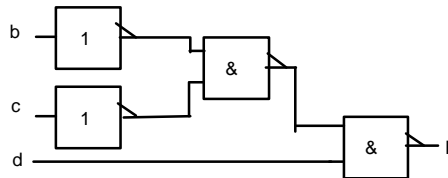


La porte 74LS00A (NAND) et l'inverseur 74ALS04B ont les valeurs extrêmes :
 $t_{PHLmax} = 8 \text{ ns}$, $t_{PHLmin} = 2 \text{ ns}$, $t_{PLHmax} = 11 \text{ ns}$ et $t_{PLHmin} = 3 \text{ ns}$.

IV - Exercices.

1) Dans un montage (ALS), une porte P alimente un circuit C dont l'entrance est de 4 charges ALS. Pour obtenir un temps de calcul meilleurs, on conserve la porte P en technologie ALS et l'on remplace le circuit C par son équivalent en technologie AS. Ce circuit AS présente des courants I_{IH} et I_{IL} de 20 microA et -1,5 mA respectivement. Vérifier le bon fonctionnement du montage.

2) Dans le schéma ci-dessous, les variables d'entrée passent de dcb=100 (F=1) à dcb=110 (F=0).
 Combien de temps met cette fonction pour passer de 1 à 0.
 (Réponse min 7 ns, max 27 ns)



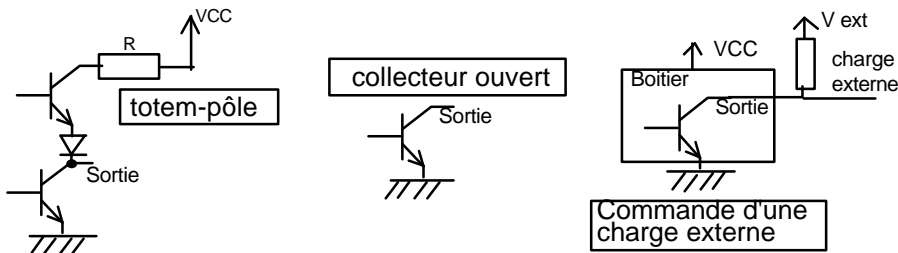
Fonction	Numéro	t_{PLH} (ns)		t_{PHL} (ns)	
		min	max	min	max
6 inverseurs	74ALS04B	3	11	2	8
4 Nand 2E	74ALS00A	3	11	2	8
3 Nand 3E	74ALS10A	2	11	2	10
2 Nand 4E	74ALS20A	3	11	3	10
1 Nand 8E	74ALS30A	3	10	3	12
4 And 2E	74ALS08	4	14	3	10
4 Nor 2E	74ALS02	3	12	3	10
3 Nor 3E	74ALS27	4	15	3	9
4 Or 2E	74ALS32	3	14	3	12
6 tampons	74ALS34	4	15	1	10

Cours 8 - Portes collecteur ouvert, trois états, trigger de schmitt.

Le but de ce TD est de présenter des portes ayant des particularités sur leur sorties ou sur leurs entrées.

I - Portes à sortie collecteur ouvert.

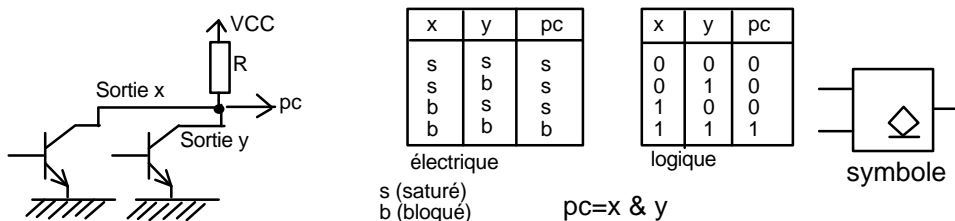
I-1) Nous présentons brièvement sur le schéma ci-dessous les différentes sorties possibles :



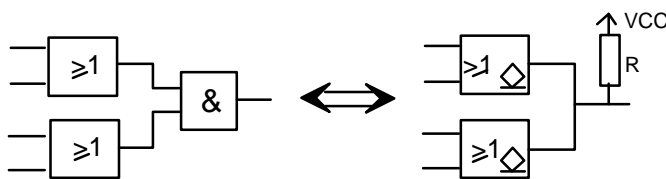
La porte à collecteur ouvert commande une charge alimentée soit sous 5 V (diode électroluminescente par exemple), soit sous une tension externe élevée, avec un courant important (relais fonctionnant sous 30 V).

I-2) mise en parallèle des sorties.

L'intérêt essentiel de tels circuits réside en fait dans la possibilité de relier entre elles les sorties de plusieurs portes à collecteur ouvert. Le point commun donne la fonction ET.



Fonction ET câblé avec collecteur ouvert



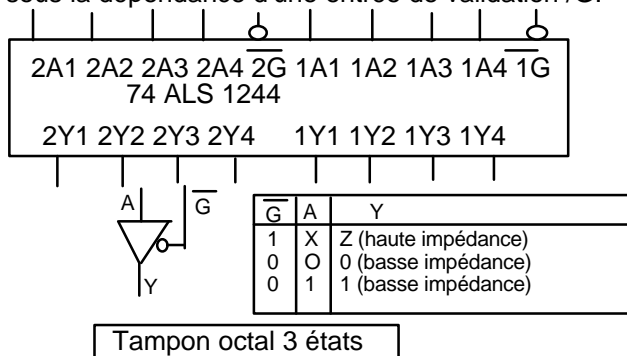
Equivalence des deux montages

I-3) paramètres électriques et temporels.

- V_{OH} : la tension de sortie au niveau haut est beaucoup plus élevée que celle d'une porte standard ; elle est égale à V_{CC} soit 5 V (contre 3 V). Ceci permet de commander des circuits ayant des tensions d'entrée V_{IH} élevées,
 - I_{OH} : le courant de sortie au niveau haut n'est pas fourni comme pour une porte standard, mais absorbé. Il vaut 0,1 mA (au lieu de 0,4 mA).
 - il n'y a pas de courant court-circuit,
 - t_{PLH} , t_{PHL} : les temps de commutation sont beaucoup plus élevés (54 et 28 ns contre 11 et 8 ns).
- Les autres paramètres sont identiques.

II - Porte à sortie 3 états.

Les portes standards ne permettent pas la mise en parallèle de leurs sorties. La solution à collecteur ouvert a l'inconvénient d'avoir des temps de commutation trop élevés. Une autre solution consiste à utiliser des portes à sortie trois états : deux de basse impédance (0 et 1) et un de haute impédance. Dans ce troisième état, tout se passe comme si les sorties étaient déconnectées. Il existe de nombreuses applications de ce type de portes, surtout lorsqu'un BUS réuni en parallèle plusieurs portes. Nous aurons l'occasion d'en voir des exemples concrets dans les TD sur le microprocesseur. Par exemple, le circuit 74 ALS 1244A comporte huit tampons trois états, répartis en deux groupes identiques placés sous la dépendance d'une entrée de validation /G.



Les paramètres électriques et temporels sont identiques sauf :

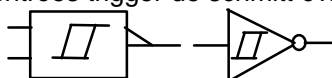
- IO_b, IO_h : courants de sortie aux niveaux bas et haut : 16 mA absorbés et 15 mA fournis, contre 8 mA absorbés et 0,4 mA fournis.
- VO_H : tension de sortie au niveau haut. Cette valeur dépend de l'importance du courant fourni IO_H ; pour une faible valeur (0,4 mA) on retrouve le VO_H d'une porte standard.
- ICC courant consommé plus important que celui d'une porte standard (20 mA au lieu de 3 mA)
- t_{PLH}, t_{PHL} : temps de propagation légèrement plus élevés que pour une porte standard (14 ns au lieu de 11 ns).
- t_{PZH}, t_{PZL}, t_{PHZ}, t_{PLZ} : ces nouveaux paramètres caractérisent le temps de passage de l'état bloqué vers les états de basse impédance (0 ou 1) et inversement.

Règles d'utilisation :

- Le point commun d'un groupement de portes à trois états ne doit pas servir à commander un autre circuit quand toutes les sorties sont dans l'état haute impédance.
- A un instant donné, une seule sortie se trouve en basse impédance dans un groupement de portes trois états. Le court-circuit entre les trois états s'appelle **conflit de bus**.

III - Portes à entrées trigger de schmitt.

Si le signal d'entrée d'une porte standard a des temps de montée et de descente trop long, la sortie oscille. Les portes à entrées trigger de schmitt évitent ce défaut



Symboles pour trigger de

Schmitt

Par rapport à une porte standard, quatre nouveaux paramètres caractérisent les entrées :

- VT₊ et VT₋ : seuils de basculement haut et bas, qui remplacent V_{IH} et V_{IL},
- IT₊ et IT₋ : courants d'entrée aux seuils haut et bas.

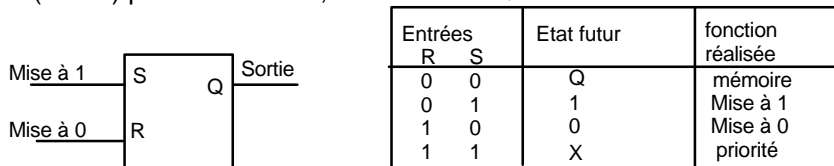
TD 6 - La fonction mémoire.

Nous allons maintenant nous intéresser à des fonctions plus complexes que les fonctions combinatoires : les fonctions séquentielles. Ces dernières sont caractérisées par le fait qu'une même combinaison des entrées peut donner des sorties différentes. En fait les sorties dépendent de l'état précédent.

I - Mémoire RS.

I-1) Principe de fonctionnement.

Ce circuit mémorise une information élémentaire (0 ou 1). Il possède une entrée S (Set) pour la mise à 1, une entrée R (Reset) pour la mise à 0, et une sortie Q donnant l'état de l'information mémorisée.

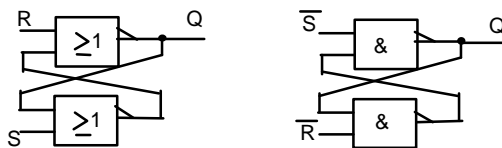


En l'absence de demande ($R=S=0$), la sortie maintient son état antérieur.

Avec les deux commandes contradictoires ($R=S=1$), la sortie de la bascule dépend de la façon dont elle est réalisée :

- si elle vaut 1, on parle de mémoire à S prioritaire,
- si elle vaut 0, on parle de mémoire à R prioritaire,
- si elle vaut l'état précédent, on parle de mémoire à mémoire prioritaire.

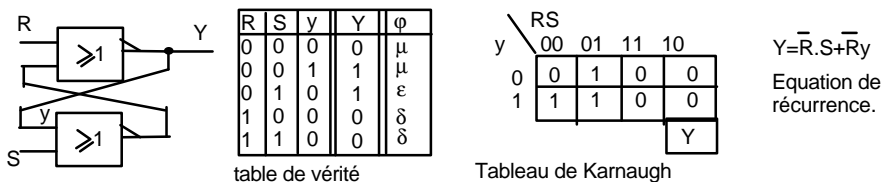
I-2) Exemples de bistable RS.



A noter que la deuxième sortie de ces montages est le complément /Q de l'autre sortie Q, sauf dans le cas $R=S=1$. On évitera donc de la noter /Q.

I-3) Méthodes d'études.

Ces circuits peuvent s'étudier, comme dans le cas du combinatoire, à l'aide des tableaux de Karnaugh (ou de tables de vérités). Mais ceux-ci étant caractérisés par un bouclage, une variable présentant l'état précédent doit y figurer. Ils sont caractérisés par une équation de récurrence.

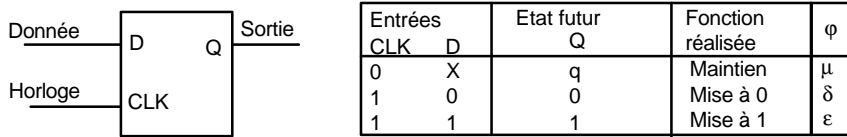


Une autre méthode consiste à les caractériser par des fonctions de commutation :

- epsilon : enclenchement (mise à 1)
- delta : déclenchement (mise à zéro)
- mu : mémoire (reste dans l'état précédent).

II - La mémoire D.

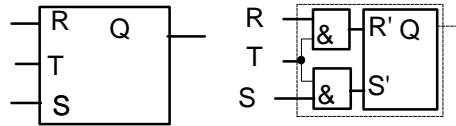
La mémoire D (D latch) possède une entrée d'horloge CLK (clock), une entrée de donnée D (Data) et une sortie Q donnant l'état de l'information mémorisée.



III - La mémoire RST.

Une entrée T supplémentaire est ajoutée à la mémoire RS servant à synchroniser les commandes R et S :

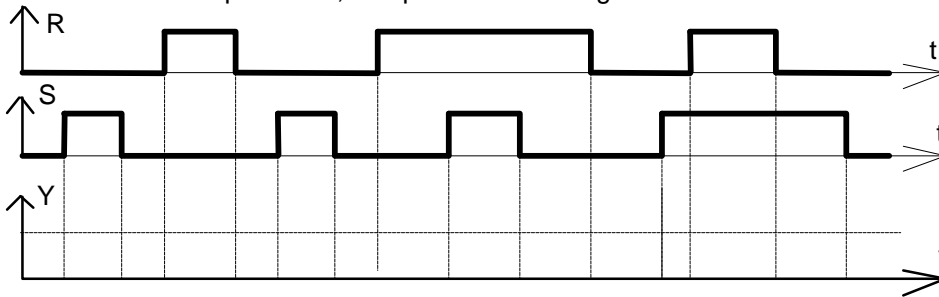
- si T=0 les entrées R et S sont inhibées,
- si T=1 les commandes sont identiques à la RS.



IV - Exercices.

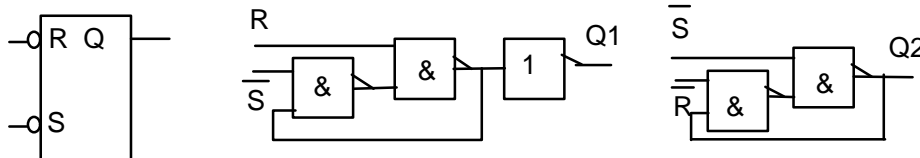
Exercice 1.

Pour une RS à mémoire prioritaire, compléter ce chronogramme.



Exercice 2.

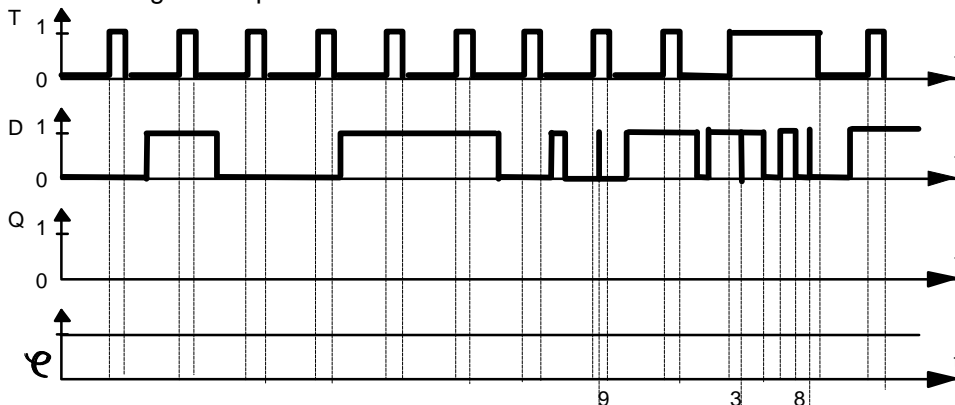
Les deux mémoires RS suivantes sont activées par des niveaux 0 :



Déterminer la nature des priorités, dessiner les en schéma croisé.

Exercice 3.

Compléter ce chronogramme pour une D latch.



Exercice 4.

Réaliser une mémoire D avec une RST.

Exercice 5.

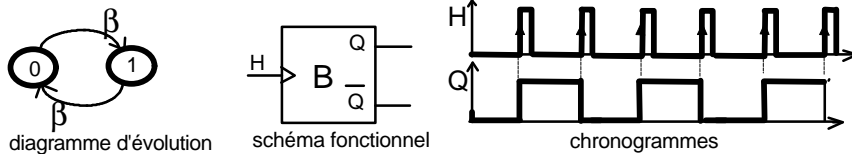
Réaliser une mémoire D autour d'une RS à ET-NON, en n'utilisant que 4 portes.

TD 7 - Les bascules.

I - Fonction bascule.

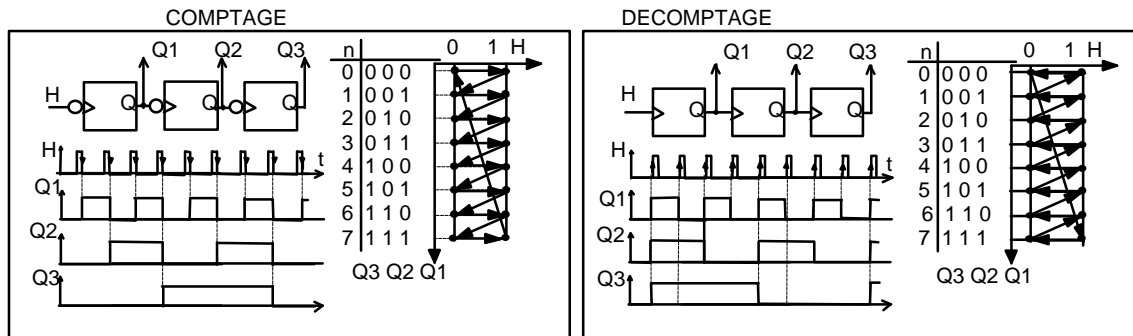
Pour leur évolution ces circuits sont pulsés par des signaux d'horloge. Ils fonctionnent sur des fronts montants ou descendants. La fonction BASCULE est telle que la sortie à l'instant $n+1$ est le complément de ce qu'elle était à l'instant n précédent. Dans ces fonctions pulsées, le temps est repéré par l'indice n , car il est maintenant discret.

Les diagrammes suivants précisent la nature d'une bascule.



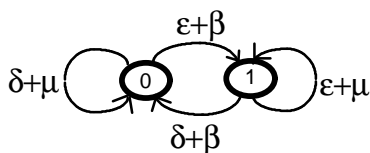
II - Comptage.

La bascule est la fonction de base des compteurs, trois bascules correspondant à trois variables d'état pouvant prendre 8 valeurs selon les combinaisons binaires. Si nous employons des bascules à front descendant, nous obtenons un compteur, et avec des fronts montants un décompteur.



III - Les différentes bascules.

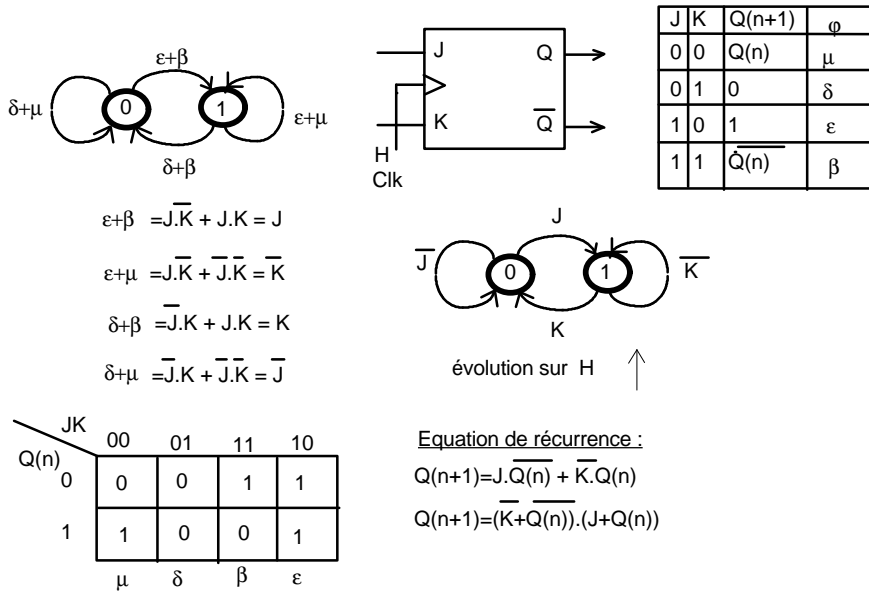
Toutes les fonctions ayant une horloge travaillant sur des fronts s'appellent "bascules". Elles ne possèdent pas toutes ce mode de commutation, mais elles peuvent toutes être câblées en bascule. Ces circuits possèdent les fonctions de commutation des mémoires (enclenchement, déclenchement et mémoire). La bascule JK rassemble toutes ces fonctions en plus de la fonction bascule.



Il ya quatre modes de commutation, donc en plus de l'horloge, deux entrées seulement suffisent pour programmer le mode de commutation désiré (ce sera la JK). Si il n'y a qu'une commande, seuls deux modes de commutation seront possibles (ce sera le cas de la D, et de la T). Si il n'y a aucune commande, seule le basculement est possible: voir ci-dessus. La JK permet de réaliser les autres.

IV - Bascule JK.

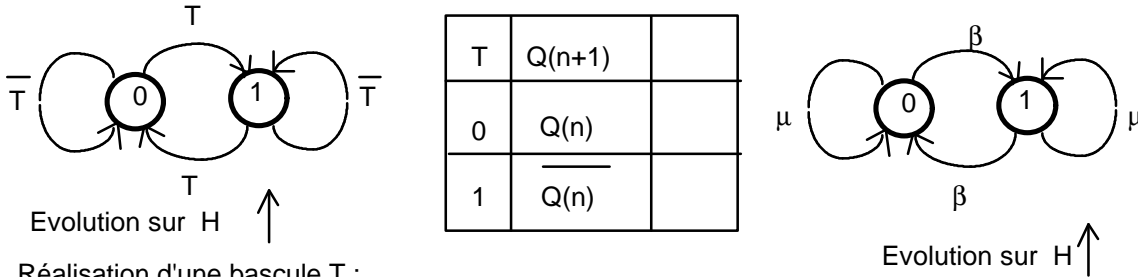
C'est la bascule la plus complète, elle utilise les quatre modes de commutation. Elle dispose de deux entrées de présélection du mode de commutation. Il existe différentes technologies de JK que nous étudierons dans un prochain chapitre. Nous considérerons ici, seulement la bascule à front positif ou négatif.



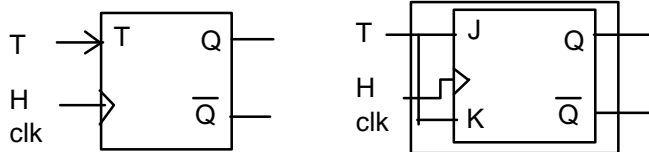
D'après la table de vérité, vérifier chaque représentation et interprétation.

V - Bascule T.

Cette bascule n'est pas commercialisée sous forme de C.I., mais on la retrouve dans certaines macrocellules de circuits programmables. Elle peut être réalisée à l'aide d'une JK. Elle utilise le mode mémoire et le mode bascule, donc une seule entrée (T) permet la sélection du mode de commutation.



Réalisation d'une bascule T :

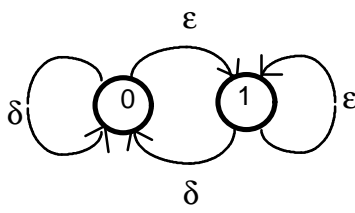


Ces bascules seront utilisées dans certaines techniques de synthèse.

VI - Bascule D.

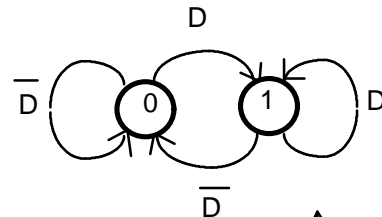
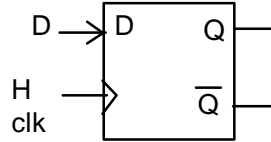
Cette bascule est réalisée en C.I., elle est très utilisée dans certaines technologies, où sa fabrication est plus facile que celle de la JK (ECL, certains réseaux programmables). Elle ne doit pas être confondue avec la mémoire D à niveau (chapitre précédent).

Elle utilise deux modes de commutation, l'enclenchement et le déclenchement. Une seule entrée (D) suffit pour choisir ce mode.



Evolution sur H ↑

D	Q(n+1)	φ
0	0	δ
1	1	ε

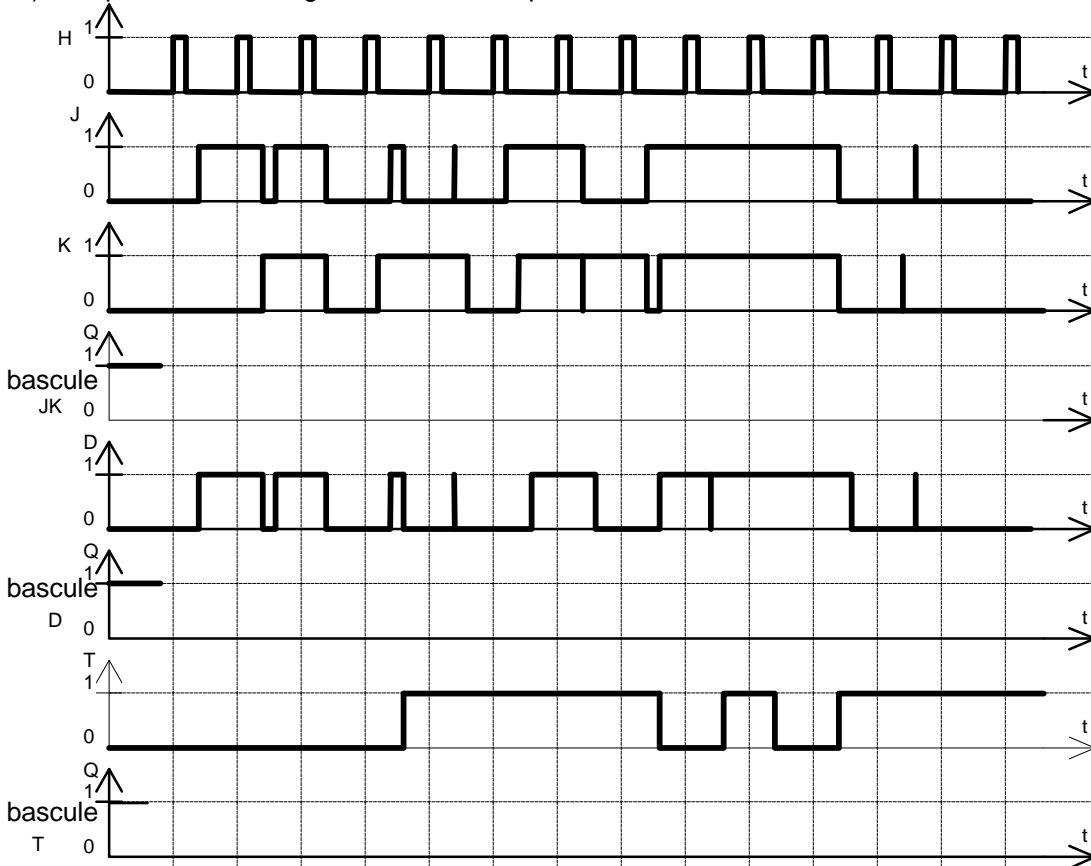


Evolution sur H ↑

Remarque : A partir d'une réflexion générale sur les quatre modes de commutation, nous voyons qu'avec une JK, nous pouvons réaliser les autres bascules (B, T, D), ce sera l'objectif d'un prochain exercice. Avec les trois autres et un environnement combinatoire, nous pouvons aussi réaliser n'importe quelle autre bascule.

VII - Exercices

1°) Compléter les chronogrammes suivants pour des bascules à front montant :



2°) Nous disposons d'une bascule JK, la transformer avec d'éventuelles portes logiques en bascules B,D, T.

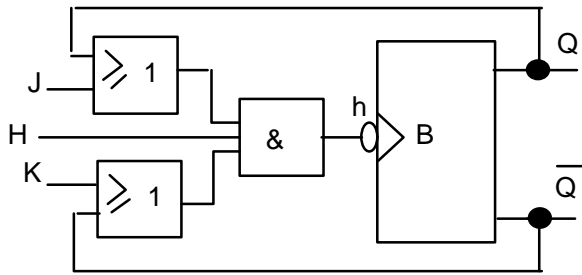
3°) Nous disposons d'une bascule D, la transformer en B, T, JK.

4°) Nous désirons réaliser la bascule suivante : E

- si E=0 Q=0,
- si E=1 $Q(n+1)=\overline{Q(n)}$ (/q : complément logique de q)

Proposer une réalisation autour d'une JK puis d'une D.

5°) Vérifier que le montage suivant réalise une JK :



TD 8 - Les registres.

Deux applications des bascules sont importantes : construction des registres et des compteurs. La propriété principale des registres est leur possibilité de stocker des données ; la propriété principale des compteurs est leur possibilité de compter, c'est à dire de décrire une séquence de nombres.

Nous allons nous intéresser maintenant aux registres. Un des plus connu est le registre à décalage. Sa principale application est de stocker des données qui arrivent, ou qui doivent être envoyées sous forme de séquence de bits. Ce registre est une machine séquentielle très simple, constitué de bascules et de portes inverseuses. Pourtant, dans la plupart des machines séquentielles une partie combinatoire conséquente est nécessaire.

I - Registres

I-1) Définitions.

Un registre est une association de mémoires unitaires, avec laquelle il est possible d'effectuer les 3 opérations fondamentales suivantes :

- **enregistrer** à un instant donné une information constituée d'une suite quelconque de 0 et 1 (mot binaire) : c'est la fonction écriture (**WRITE**) ou chargement (**LOAD**)
- **conserver** ce mot en mémoire aussi longtemps que nécessaire : fonction mémorisation (**STORAGE**)
- **restituer** le mot enregistré à la demande : fonction lecture (**READ**).

D'autres types de circuits permettent d'effectuer ces mêmes opérations, mais sur plusieurs mots qui sont alors classés et identifiés à l'aide d'une adresse : ce sont les mémoires actives (**ACTIVE MEMORIES**) appelées aussi R.A.M (**RANDOM ACCES MEMORIES**). Il existe aussi des mémoires mortes R.O.M (**READ ONLY MEMORIES**) pour lesquelles la fonction écriture a été réalisée une fois pour toutes lors de la fabrication.

I-2) Structures des registres.

La structure d'un registre dépendra du mode, **série ou parallèle**, utilisé :

- pour y écrire l'information,
- pour la lire ensuite.

On distinguera donc, des registres :

- à écriture et lecture en parallèle, (registre tampon, **BUFFER REGISTER**)
- à écriture et lecture en série, (registre à décalage, **SHIFT REGISTER**)
- à écriture en parallèle et lecture en série, (**PARALLEL IN- SERIAL OUT**)
- à écriture en série et lecture en parallèle. (**SERIAL IN - PARALLEL OUT**)

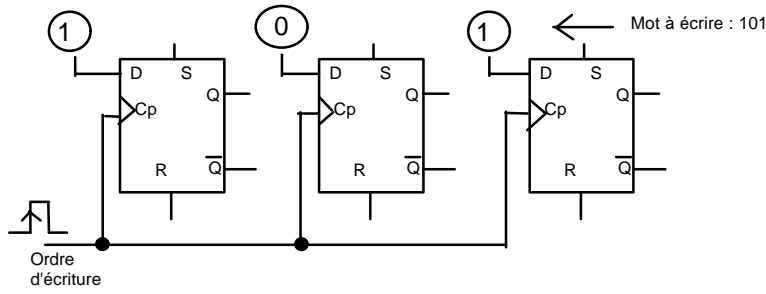
I-3) Ecriture d'un mot dans un registre.

Quelle que soit la structure du registre chaque bit du mot est toujours écrit et gardé en mémoire grâce à une fonction D.

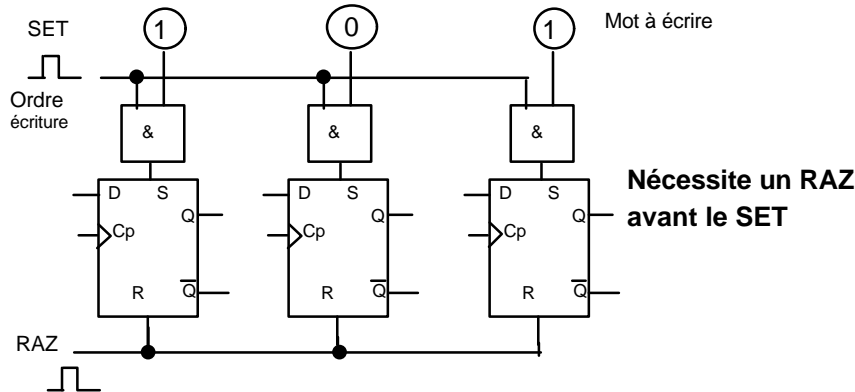
I-3-a) écriture en parallèle

peut se faire de 2 manières :

- soit sur l'entrée D des bascules (entrée synchrone),



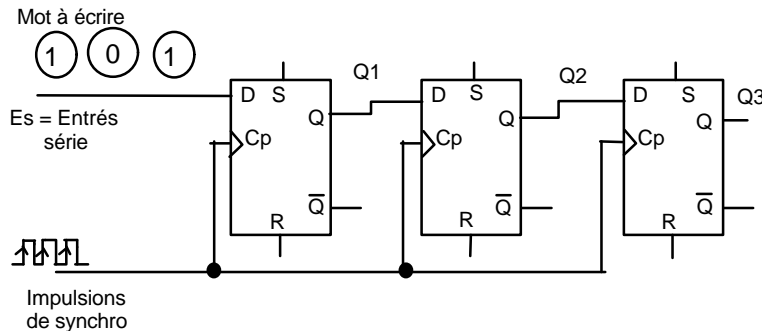
- soit sur les entrées de forçage (entrées asynchrones).



I-3-b) écriture en série

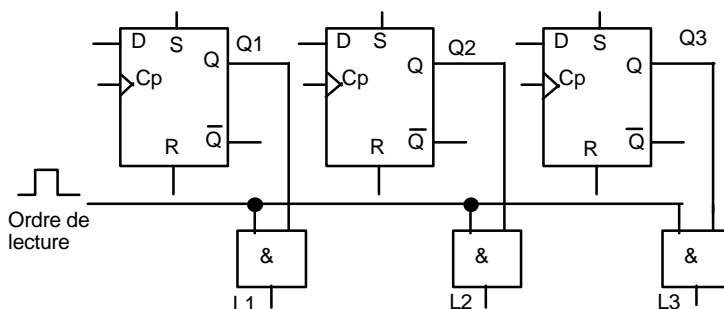
Pour écrire en série un mot de 3 bits (par exemple), dans un registre de 3 bascules, il faut 3 ordres d'écriture. Il faut donc :

- que la première bascule recopie à chaque ordre d'écriture la valeur logique présente sur l'entrée série du registre,
- que la seconde recopie à chaque ordre d'écriture l'état dans lequel la première, juste avant cet ordre,
- la troisième bascule recopie de la même manière les états successifs de la seconde.

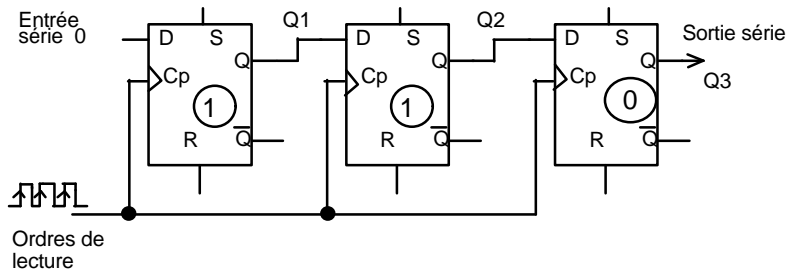


I-4) Lecture d'un mot contenu dans un registre.

I-4-a) lecture en parallèle



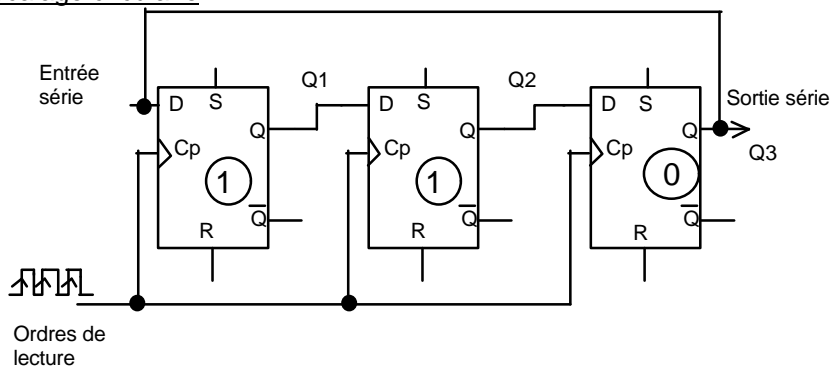
I-4-b) lecture en série simple.



I-4-c) lecture en série répétitive.

Dans les deux exemples précédents, le mot contenu dans le registre se trouve, après lecture en série, remplacé par un autre (000 dans l'exemple choisi). Il est donc détruit.

Registre à décalage circulaire :



II - Exercices.

1°) Donner le schéma complet d'un registre de 3 mémoires à écriture et lecture en parallèle, réalisé avec des mémoires D de type LATCH. Précisez l'allure du signal de commande d'écriture. Peut-on réaliser un registre à écriture et lecture série avec des mémoires D ?

2°) Donner le schéma d'un registre de 3 bascules à écriture et lecture en série réalisé avec des bascules JK.

3°) Donner la liste des 5 modes de fonctionnement du registre de décalage 74194.

La seule entrée asynchrone du 74194 qui est prioritaire sur toutes les autres entrées est _____.

Quel est l'effet d'une impulsion d'horloge quand le registre 74194 est dans l'état d'attente passive ?

Il faut _____ impulsion(s) d'horloge pour le chargement parallèle de 4 bits dans le 74194.

L'entrée D_{SR} est en fonctionnement pour $S_0 = \underline{\quad}$ (0) (1) et $S_1 = \underline{\quad}$ (0) (1).

Le 74194 est à activation par _____ (front positif d'impulsion) (impulsion).

4 - BIT BIDIRECTIONAL UNIVERSAL SHIFT REGISTER

54/74 SERIES "194"

FUNCTIONAL DESCRIPTION

The functional characteristic of the "194" 4-Bit Bidirectional Shift Register are indicated in the Logic Diagram and Truth Table. The register is fully synchronous, with all operations taking place in less than 20 nanoseconds (typical) for the 54/74 and 54LS/74LS, and 12 ns (typical) for 54S/74S, making the device especially useful for implementing very high speed CPUs, or for memory buffer register.

The "194" design has special logic features which increase the range of application. The synchronous operation of the device is determined by two Mode Select inputs, S_0 and S_1 . As shown in the mode Select Table, data can be entered and shifted from the left to right (shift right, $Q_0 \rightarrow Q_1$, etc.) or ; right to left (shift left, $Q_3 \rightarrow Q_2$, etc.) or, parallel data can be entered loading all four bits of the register simultaneously. When both S_0 and S_1 are LOW, existing data is retained in a hold (do nothing) mode. The first and last stage provide D-type serial data inputs (D_{SR}, D_{SL}) to allow multistage shift right or shift left data transfers without interfering with parallel load operation.

Mode Select and Data inputs on the 54S/74S194 and 54LS/74LS194A are edge triggered, responding only to the LOW-to-High transition of the clock (CP).

MODE SELECT - FUNCTION TABLE

OPERATING MODE	INPUTS							OUTPUTS			
	CP	MR	S_1	S_0	D_{SR}	D_{SL}	D_n	Q_0	Q_1	Q_2	Q_3
Reset (clear)	X	L	X	X	X	X	X	L	L	L	L
Hold (do nothing)	X	H	l(b)	l(b)	X	X	X	q0	q1	q2	q3
Shift Left	↑	H	h	l(b)	X	l	X	q1	q2	q3	L
Shift Left	↑	H	h	l(b)	X	h	X	q1	q2	q3	H
Shift Right	↑	H	l(b)	h	l	X	X	L	q0	q1	q2
Shift Right	↑	H	l(b)	h	h	X	X	H	q0	q1	q2
Parallel Load	↑	H	h	h	X	X	dn	d0	d1	d2	d3

SIGNETICS

H = HIGH voltage level

h = HIGH voltage level one setup time prior to the LOW-to-HIGH clock transition

L = LOW voltage level

l = LOW voltage level one setup time prior to the LOW-to-HIGH clock transition

dn (qn) = Lower case letters indicate the state of the referenced inputs (or outputs) one setup time prior to the LOW-to-HIGH clock transition

X = Don't care

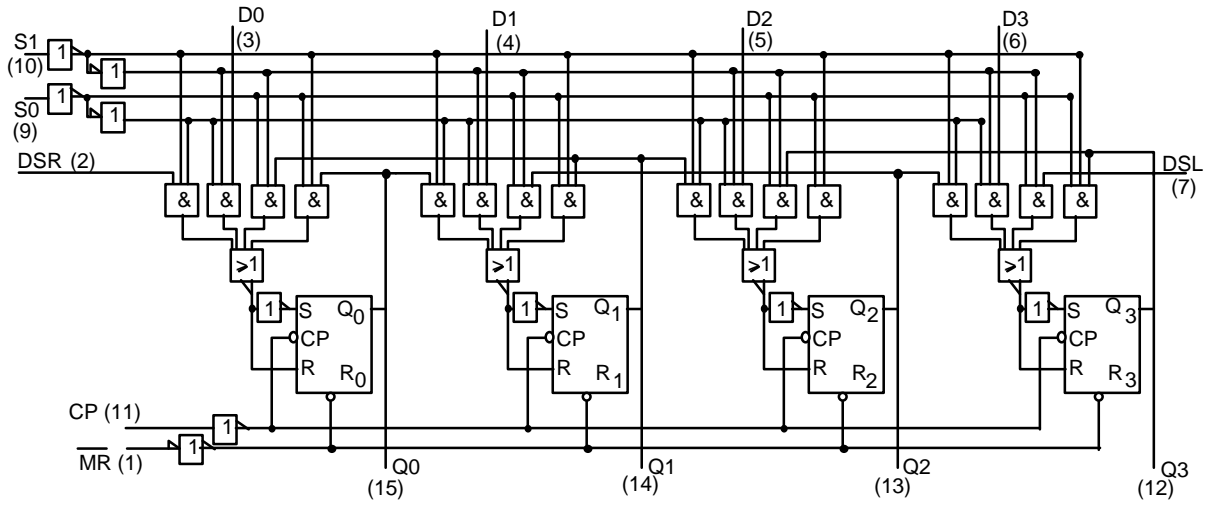
↑ = LOW-to-HIGH clock transition

NOTES

b. the HIGH-to-LOW transition of the S_0 and S_1 inputs on the 54/74194 should only take place while CP is HIGH for conventional operation.

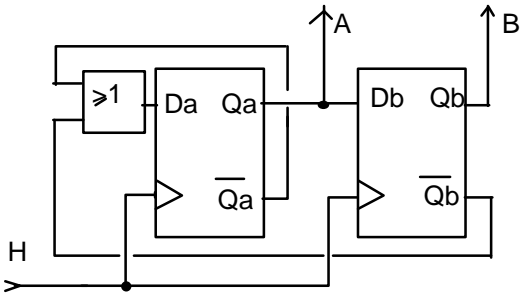
Therefore, the only timing restriction is that the mode Control and selected Data Inputs must be stable one setup time prior to the positive transition of the clock pulse.

The four parallel inputs ($D_0 - D_3$) are D-type inputs. data appearing on $D_0 - D_3$ inputs when S_0 and S_1 are HIGH is transferred to the $Q_0 - Q_3$ outputs respectively following the next LOW-to-HIGH transition of the clock. When LOW, the asynchronous Master reset (/MR) overrides all the other input conditions and forces the Q outputs LOW.



4°) Donner le schéma d'un registre de 3 bits, programmable, à écriture et lecture en série par décalage à droite ou à gauche, circulaire ou non. Prévoir 2 entrées de programmation P1 et P2, et donner le code de programmation choisi. Utiliser des bascules D synchrones à front montant.

Indications : on a encore ici une bonne illustration de l'utilisation de la méthode du SI-ALORS.

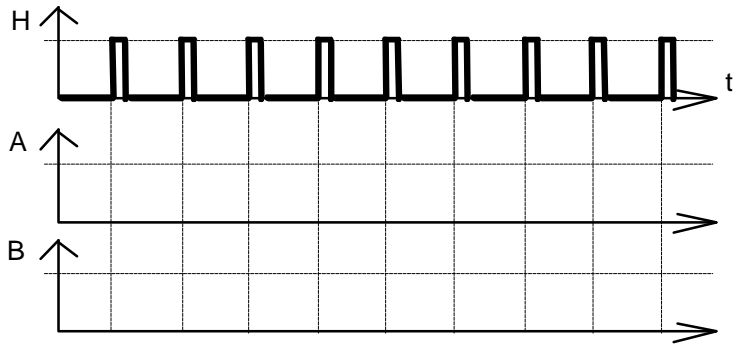


Da = B

Db = A

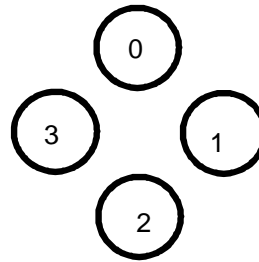
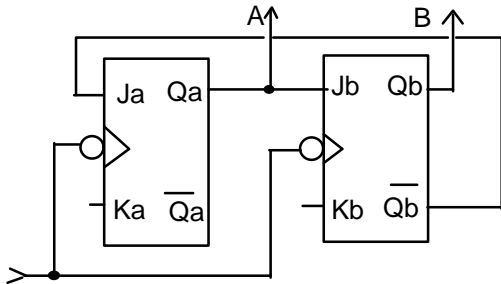
2 variables -> 4 états.

Si état actuel	Alors				Alors état futur
	B	A	Db	Da	
0					
1					
2					
3					

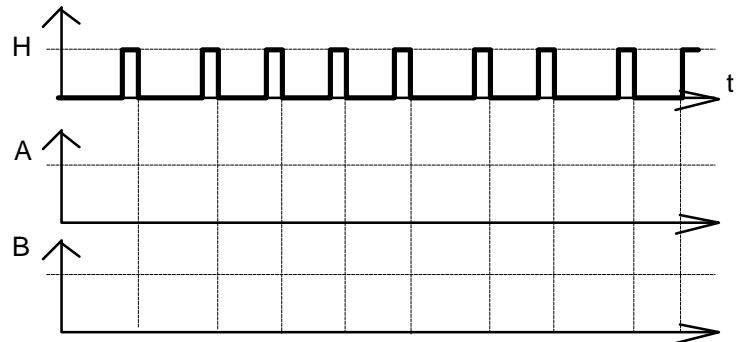


I-5) Montage avec JK.

Cette même méthode peut s'appliquer aux montages à JK.



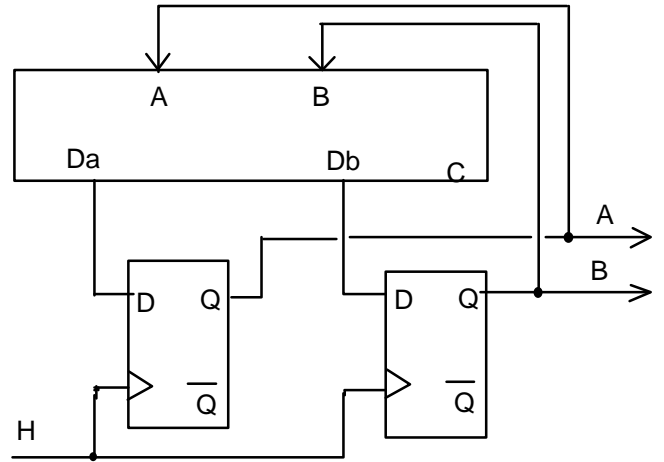
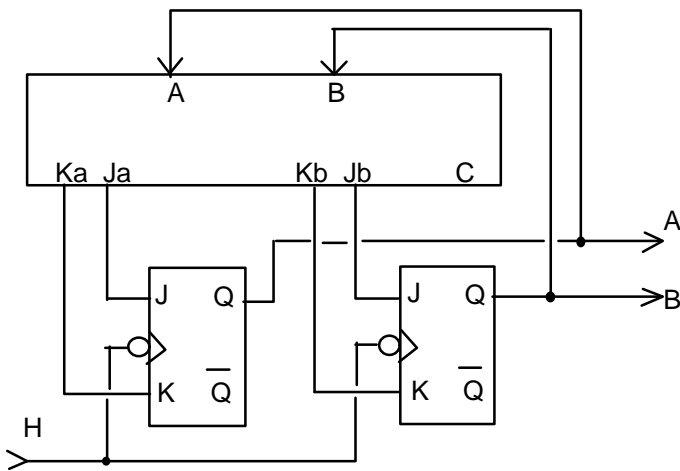
Si n	Alors						Alors état futur				
	B	A	Jb	Kb	φ _b	Ja		Ka	φ _a	B	A
0											
1											
2											
3											



II - Synthèse des compteurs synchrones.

Il s'agit de rechercher les lois de commande des entrées afin de présélectionner la fonction de commutation désirée. Il s'agit donc d'une recherche de type combinatoire. la synthèse à JK pourrait paraître plus compliquée qu'avec des D. Mais ces bascules offrant plus de possibilités amènent à des schémas plus simples.

Exemple de structure pour deux variables d'état :



Synthèse à JK : recherche de A fonctions

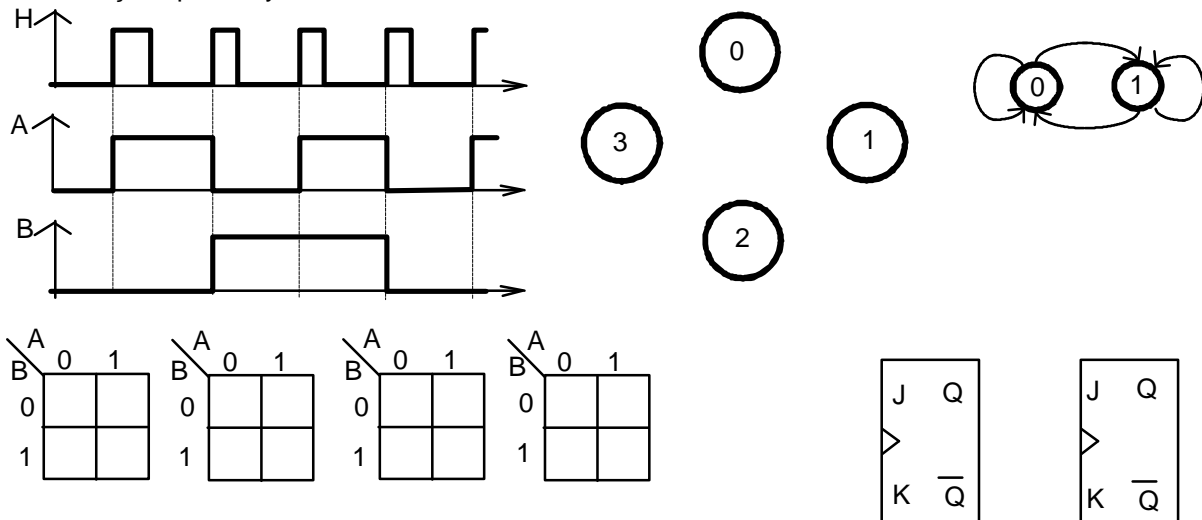
à D : recherche de 2 fonctions.

Synthèse d'un compteur modulo 4.

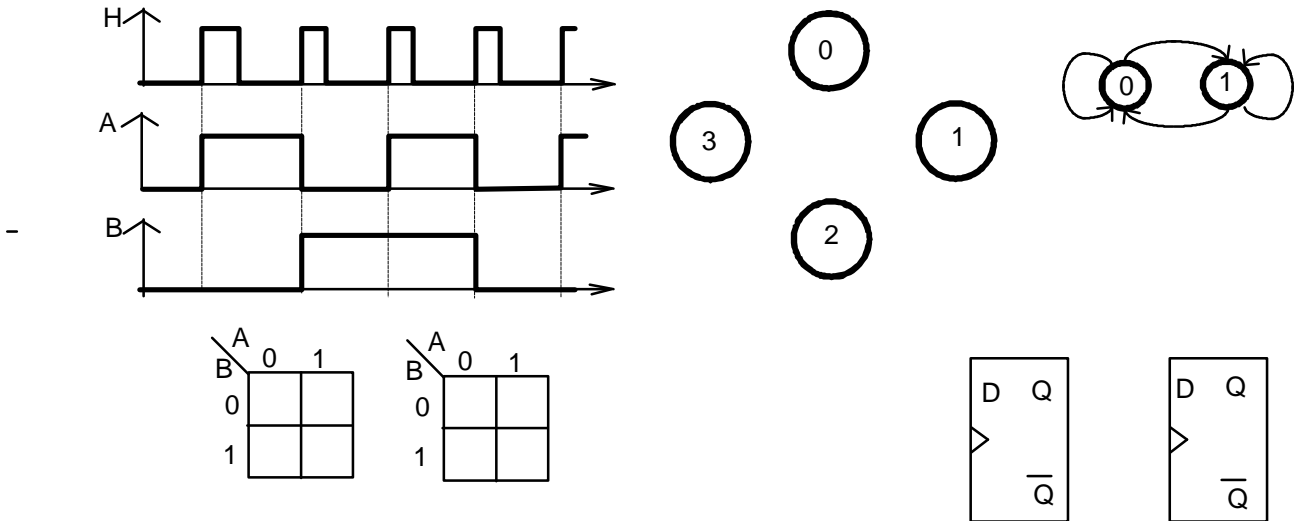
Note : le terme modulo provient de l'arithmétique entière, où le modulo est le reste de l'opération de division (ex: 18 modulo 4 = 2). Il en est de même pour un compteur : 18 impulsions à compter en partant de 0 à l'entrée d'un compteur par 4, il arrivera à l'état 2.

On a besoin de 4 états pour réaliser cela donc de 2 bascules.

Commençons par la synthèse en JK :



Réalisez le même compteur modulo 4 avec deux bascules D.



Vous devez constater une complication du schéma par rapport aux JK.

III - Exercices.

1°) Réalisez la synthèse d'un diviseur de fréquence par trois à JK et à D. Un diviseur de fréquence n'est pas un compteur, il sert en électronique pour l'élaboration de fréquence. Vous pouvez choisir les cycles que vous voulez. Il n'y a pas de considération concernant l'état initial. Par contre toutes les phases devront être connexes.

2°) Réalisez un générateur de signaux carrés déphasés de 90° (en quadrature)

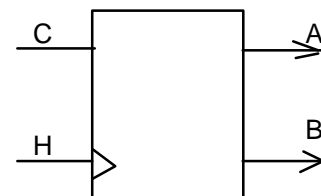
3°) Réaliser un générateur de signaux triphasés (120°). Le rapport cyclique sera de 1/2. Montrer qu'il faut nécessairement 6 états pour le réaliser.

4°) Nous allons étudier une machine possédant une entrée autre que H. Nous appellerons cette entrée C.

- Si C=0 on a le cycle : ① → ③ ← ②
- Si C=1 on a le cycle : ① → ② ← ③

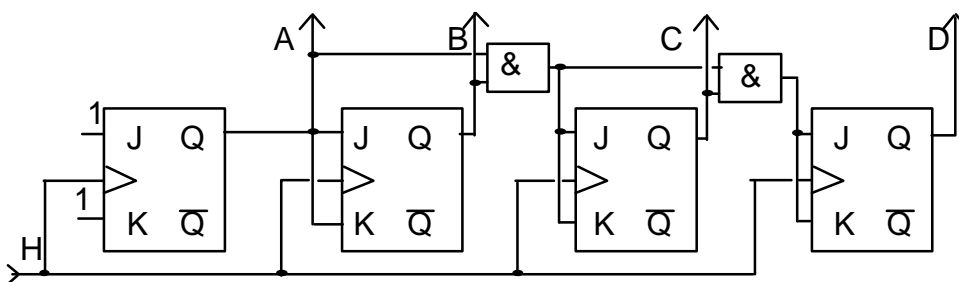
Les changements de cycle selon C se feront en synchronisme avec l'horloge.

Dans cet exercice, il est important de travailler avec le diagramme des phases et non pas avec les chronogrammes qui sont trop complexes.

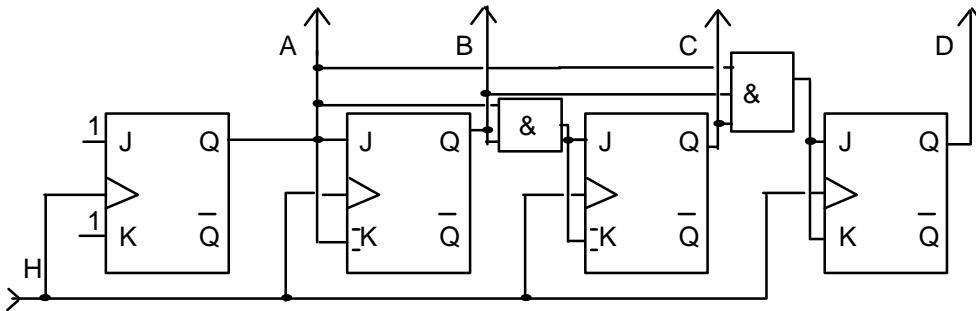


IV - Limitation en fréquence des compteurs synchrones.

Ce montage réalise un compteur binaire modulo 16. Il est limité en fréquence par les temps de propagation (t_p) des portes et des bascules.



Pour un temps t_p moyen de 20 ns par porte et par bascule, quelle serait la fréquence maximum applicable en entrée pour un fonctionnement correct.
Même question pour le schéma ci-dessous :



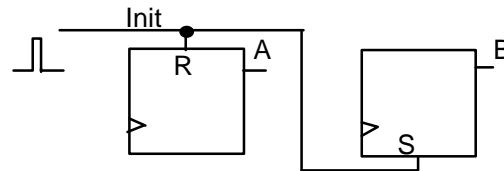
Généralisation et conclusion : ?

V - Initialisation d'un compteur.

Pour initialiser (première phase) ou pour éviter des cycles parasites, il faut parfois imposer les conditions de départ par un forçage asynchrone en utilisant les R et S des bascules :

ici : départ $A=0, B=1$.

Proposez un schéma RC pour avoir une initialisation à la mise sous tension sur les /R et /S.



VI - Forçage asynchrone des compteurs synchrones (à rétroaction).

Ce type de montage se rencontre parfois dans la pratique, pour des cas simples. Il est cependant déconseillé de l'employer.

Exemple : transformer un compteur modulo 16 en diviseur de fréquence par 12 (comptage de 0 à 11).

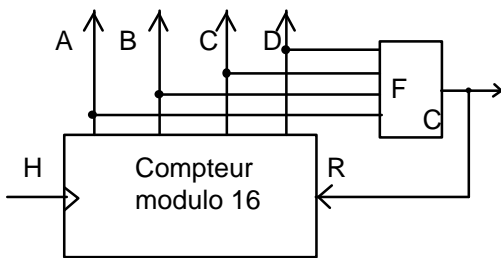
SI (n=12) ALORS n := 0

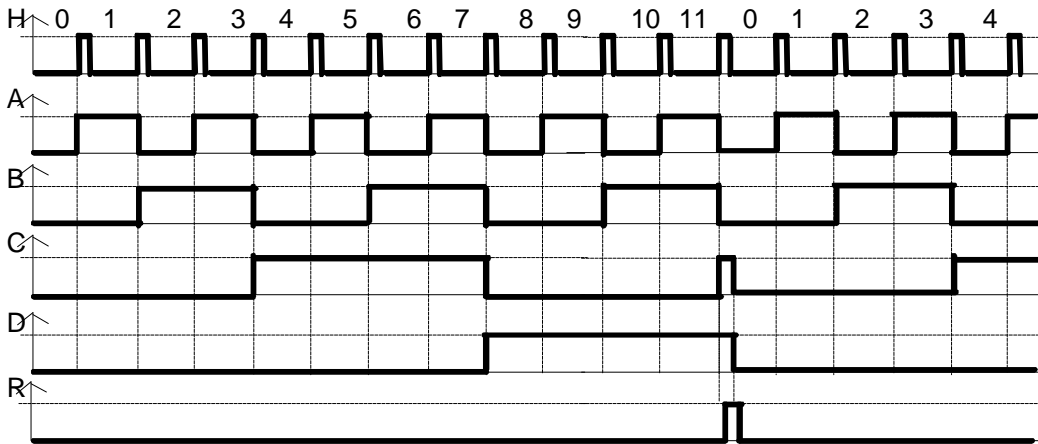
$n = 12$ si $\overline{A} \cdot \overline{B} \cdot C \cdot D = 1$

$R = F(A, B, C, D)$

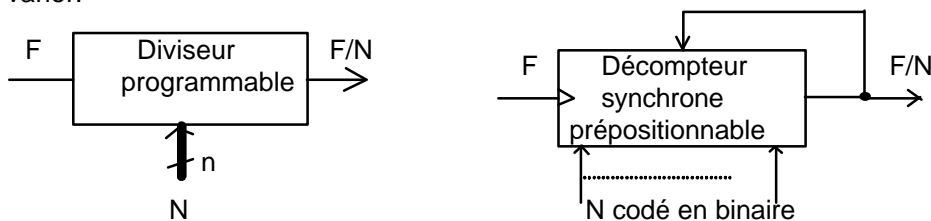
$R = C \cdot D$

Pourquoi cette simplification ?





Exemple d'application complexe : le diviseur programmable. Le nombre N est codé en binaire, il peut varier.



VII - Compteurs et le langage ABEL.

Il y a trois manières de spécifier un fonctionnement séquentiel de type compteur :

- équations de récurrences (equations)
- diagramme de transition (state_diagram)
- tables de vérités (truth_table)

Nous donnons ci-dessous un exemple pour lequel il y a une partie séquentielle et une partie combinatoire.

PARTIE COMMUNE AUX TROIS PROGRAMMES

equations

```
etat.clk = ck; "declare qui
               "est horloge
etat := etat + 1; "equation
               "sequentielle
out = q1 $ q0; "equation
               "combinatoire
```

PARTIE COMMUNE AUX TROIS PROGRAMMES

```
MODULE EXO8
title 'exercice 8 compteur binaire 4 etats'
ck pin; "entree horloge
q1,q0 pin istype 'reg'; "sorties sequentielles
out pin istype 'com'; "sortie combinatoire
X=.X.;
Xs=[X,X,X];
etat = [q1, q0];
outs=[q1,q0,out];
front =.c.;
```

```
equations
etat.clk = ck;
out = q1 $ q0; "combinatoire
state_diagram etat
state [0,0] : goto [0,1];
state [0,1] : goto [1,0];
state [1,0] : goto [1,1];
state [1,1] : goto [0,0];
```

```
test_vectors (ck -> outs)
front -> Xs;
front -> Xs;
front -> Xs;
front -> Xs;
front -> Xs;
```

end

equations

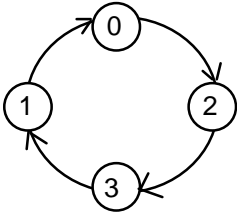
```
etat.clk = ck;
truth_table (etat.fb->etat->out)
0 :> 1 -> 0; "specification
1 :> 2 -> 1; "decimale
2 :> 3 -> 1;
3 :> 0 -> 0;
```

Remarques : Notez que lors de la spécification par table de vérité, la partie combinatoire se spécifie sur l'état présent (noté .fb).

La spécification par diagramme d'état pourrait elle aussi se faire en décimal.

Les équations de récurrence pourraient être détaillées sous la forme : $q1 := \dots$ et $q0 := \dots$ qui seraient des équations logiques. Le + utilisé ici n'est pas un OU logique mais bel et bien une addition binaire.

Il est bien évident que la spécification par diagramme d'état est la plus simple :



"s'écrit tout simplement :

etat = [q1, q0]; "q0 poids faible car a droite

.....

equations

etat.clk = ck;

state_diagram etat

state 0 : goto 2;

state 2 : goto 3;

state 3 : goto 1;

state 1 : goto 0;

Remarque : On ne peut pas se passer de la partie equations ! qui renseigne ABEL sur l'horloge.

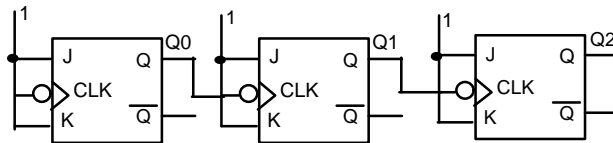
TD 10 - Compteurs asynchrones.

Nous allons dans ce cours examiner rapidement le comptage asynchrone. Il est caractérisé par le fait que toutes les bascules utilisées n'ont pas une horloge commune.

Nous allons d'abord nous intéresser à l'analyse de ces compteurs sur des circuits existants, puis nous proposerons à titre d'exercice un problème de synthèse. Ce problème ne sera résolu en TD que si le temps le permet.

I - Analyse de circuits existants.

I-1) Présentation générale : le compteur binaire.



Compteur asynchrone avec bascules JK

On a souvent besoin de compteurs modulo N c'est à dire comptant entre 0 et N-1. Pour réaliser un tel compteur il existe deux solutions :

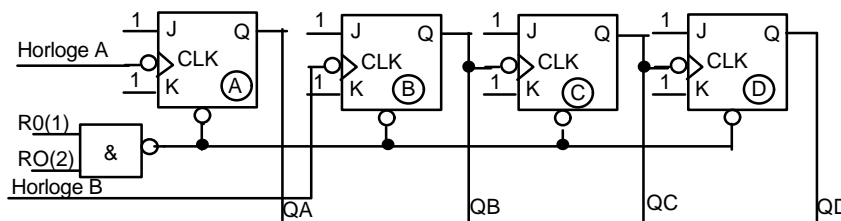
- un bouclage asynchrone,
- un conditionnement des entrées des bascules.

I-2) Le compteur 7493.

Ce circuit est composé de 4 bascules JK. Il possède 4 commandes : 2 horloges et 2 commandes de remise à 0.

Si l'on n'utilise que l'entrée A d'horloge on aura un compteur modulo 2.

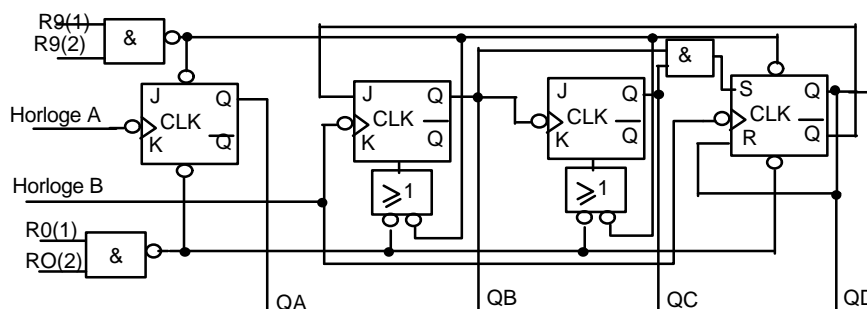
En utilisant l'entrée d'horloge B et les sorties Q_B, Q_C, Q_D on a un compteur modulo 8.



I-3) Le compteur 7490.

Ce circuit est composé d'un diviseur par 2 et d'un diviseur par 5. Par mise en cascade, il permet une division par 10. Le cycle à 10 positions est différent suivant l'ordre de la mise en cascade. Le cycle ayant un rapport cyclique de 1/2 est appelé Bi-quinaire, et l'autre comptage BCD.

Il comprend d'autre part une entrée de mise à 0 et une entrée de mise à 9.



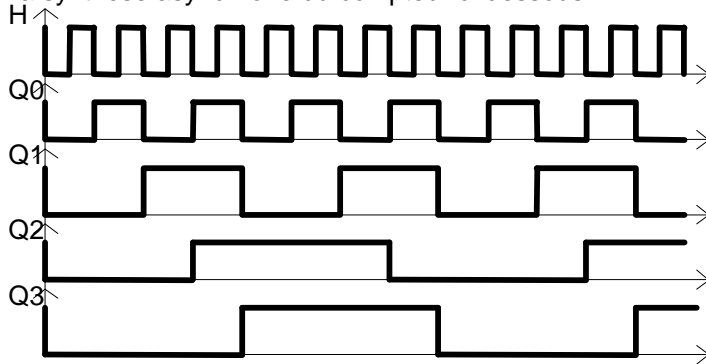
II - Exercice.

A l'aide de la structure interne du 7490 ci-dessus, on vous demande de remplir les deux tableaux des séquences. Trouver lequel correspond au comptage BCD et lequel au Bi-quinaire ?

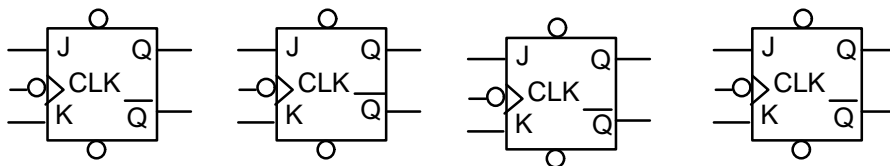
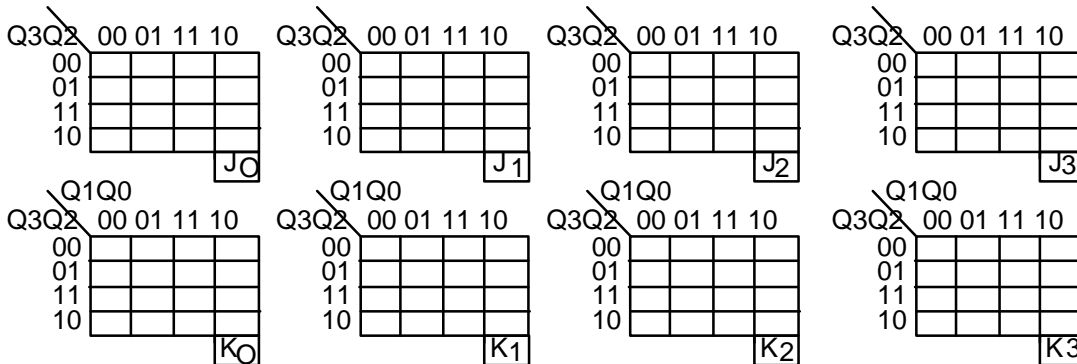
Etat	Sorties				Etat	Sorties			
	QD	QC	QB	QA		QD	QC	QB	QA
0					0				
1					1				
2					2				
3					3				
4					4				
5					5				
6					6				
7					7				
8					8				
9					9				

III - Synthèse.

Réaliser la synthèse asynchrone du compteur ci-dessous.



Remplir les tableaux de Karnaugh et en déduire le schéma.



NOM :

Prénom :

(31 Janvier 1996)

Devoir Surveillé d'informatique industrielle n°2

Feuille Réponse n°1

Exercice n°1.

On désire réaliser la synthèse d'un compteur sur 3 bits caractérisé par le chronogramme ci-dessous. On désire utiliser 3 bascules JK pour réaliser ce compteur. Compléter les diagrammes de transition (ou d'évolution) puis les tableaux de Karnaugh.

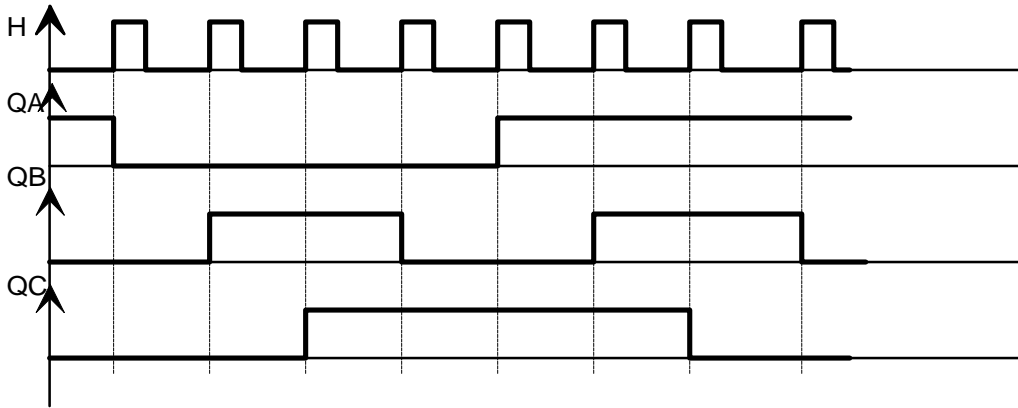


Diagramme de transition de la JK	Diagramme de transition à compléter								
<table border="1" style="margin: auto;"> <tr> <th>J</th> <th>K</th> <th>Q(n+1)</th> <th>j</th> </tr> <tr> <td> </td> <td> </td> <td> </td> <td> </td> </tr> </table> <p>Table de vérité</p> <p>Diagramme d'évolution A COMPLETER</p>	J	K	Q(n+1)	j					<p style="text-align: center;">Qc, Qb, Qa</p>
J	K	Q(n+1)	j						

		QbQa				
		00	01	11	10	
Qc	0					Jc
	1					

		QbQa				
		00	01	11	10	
Qc	0					Jb
	1					

		QbQa				
		00	01	11	10	
Qc	0					Ja
	1					

		QbQa				
		00	01	11	10	
Qc	0					Kc
	1					

		QbQa				
		00	01	11	10	
Qc	0					Kb
	1					

		QbQa				
		00	01	11	10	
Qc	0					Ka
	1					

NOM :

Feuille réponse n°2

Prénom :

En déduire les équations

Jc =

Kc =

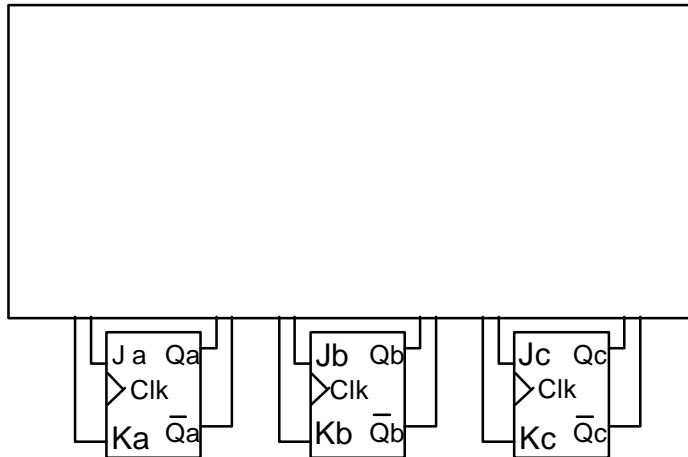
Jb =

Kb =

Ja =

Ka =

Compléter le schéma ci-contre y compris le signal d'horloge : la synthèse sera faite avec les portes de votre choix.



Exercice 2.

On souhaite effectuer la conversion d'un mot de données de 8 bits arrivant par le bus de données B₁ sous forme parallèle, en un mot de données 8 bits repartant sous forme série sur B₂. Pour cela on utilise le montage décrit par la figure 1.

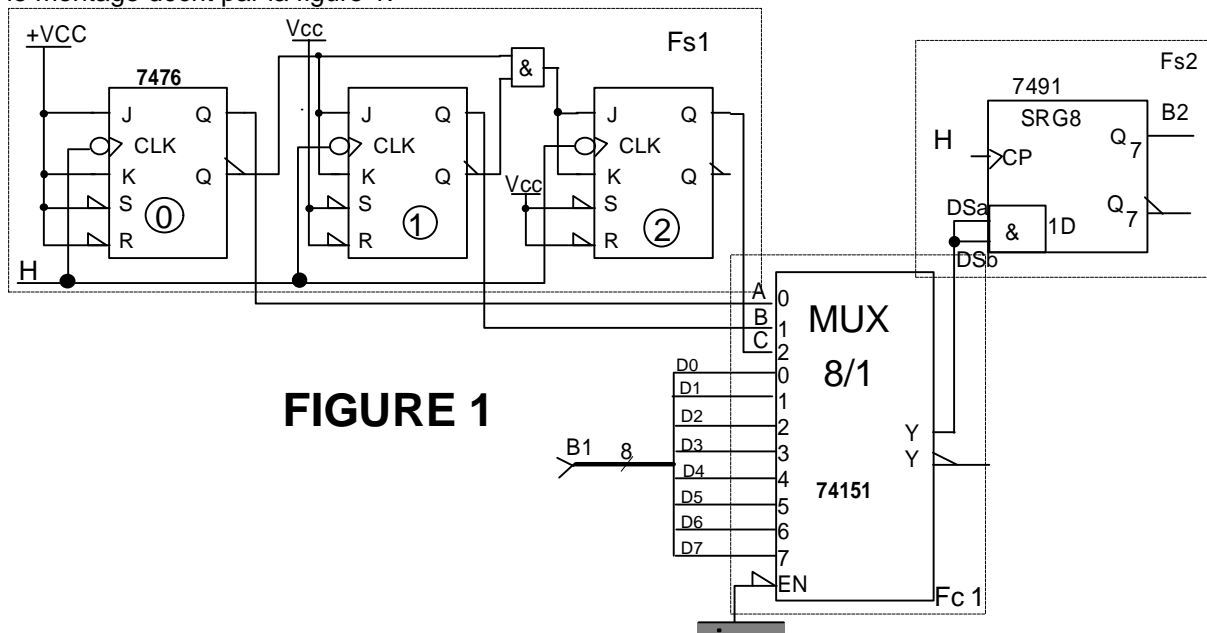


FIGURE 1

1°) La transformation du mot parallèle en série est assurée par le multiplexeur 8 vers 1 (Cl₃ - doc 1). La sélection des données est assurée par le compteur/décompteur Fs₁.

a) A quoi servent les entrées S et R des bascules JK de la figure 1 ? Appelle-t-on cela des entrées synchrones ou asynchrones. Précisez le front d'horloge actif en H.

Réponse :

b) Analyser le schéma de la fonction fs₁ (figure 1) et donner les équations des entrées J et K de chaque bascule.

Réponse :

J ₀ =	J ₁ =	J ₂ =
K ₀ =	K ₁ =	K ₂ =

NOM :

Feuille réponse n°3

Prénom :

c) Remplir la table de vérité ci-dessous définissant le cycle de fonctionnement de fs_1 puis précisez la fonction de comptage obtenue.

Etat actuel			Entrées			Etat futur					
Q_2	Q_1	Q_0	K_2	J_2	K_1	J_1	K_0	J_0	Q_2	Q_1	Q_0

Réponse type de comptage :

- type :
- modulo :

d) Représenter l'évolution temporelle des sorties Q_0 Q_1 Q_2 du compteur/décompteur sur le chronogramme figure 2.

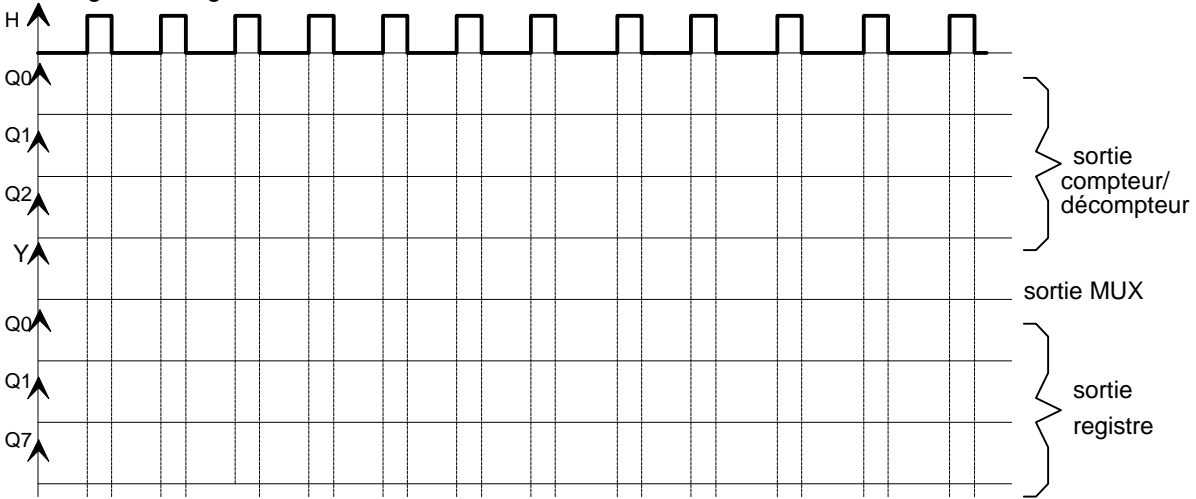


FIGURE 2

2°) On entre sur le bus de données B_1 un mot binaire $D_7D_6...D_0$ égal à $(10111010)_2$.

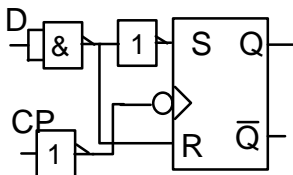
a) Analyser le schéma de la fonction F_{c1} puis préciser le rôle et le niveau d'activation

- de l'entrée /G
- des entrées C, B, A.

Réponses :

b) Si $Q_0=A$, $Q_1=B$ et $Q_2=C$, compléter sur le chronogramme de la figure 2 la représentation de l'évolution temporelle de la sortie Y.

3°) La mémorisation du mot binaire apparaissant en Y nécessite le registre défini par la fonction F_{s2} sur le schéma figure 1.



a) Donner la table de vérité de la sortie Q en fonction de D et CP pour la bascule définie ci-contre en précisant le front actif de l'horloge sur CP et les modes de fonctionnements obtenus. En déduire le graphe d'évolution (graphe des états ou graphe de transitions)

CP	D	$Q(n+1)$	j : mode

Table de vérité



Diagramme d'évolution

NOM :

Prénom :

Feuille réponse n°4

b) En exploitant la documentation du 7491 (doc 3) préciser de quel type de registre il s'agit et sa dénomination anglaise.

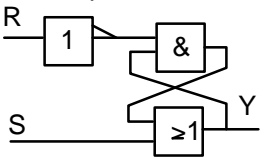
Réponses :

c) Si $C_p = H$, représenter sur la figure 2 l'évolution temporelle des sorties Q_0 , Q_1 et Q_7 de ce registre. Préciser le nombre d'impulsions d'horloge H nécessaire pour obtenir sur B_2 , la sortie complète du mot binaire entré sur B_1 .

Réponse :

Exercice 3. (mémoire)

On utilisera la notation habituelle qui consiste à utiliser les minuscules pour les variables avant changement d'état et les majuscules pour les variables après changement.

	Compléter tableau de Karnaugh	Equation simplifiée de Y																				
<p>Une fonction mémoire est décrite par son schéma :</p> 	<p>RS00 01 11 10</p> <table border="1" data-bbox="510 1030 821 1209"> <tr> <td>y</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>0</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>1</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td>Y</td> </tr> </table>	y					0					1									Y	<ul style="list-style-type: none"> • sous forme disjonctive <u>Réponse :</u> • sous forme conjonctive <u>Réponse :</u>
y																						
0																						
1																						
				Y																		
<p>Quelle est la priorité ?</p> <p><u>Réponse :</u></p>	<p>Faire un schéma croisé en Nands</p>	<p>Faire un schéma croisé en NORs</p>																				

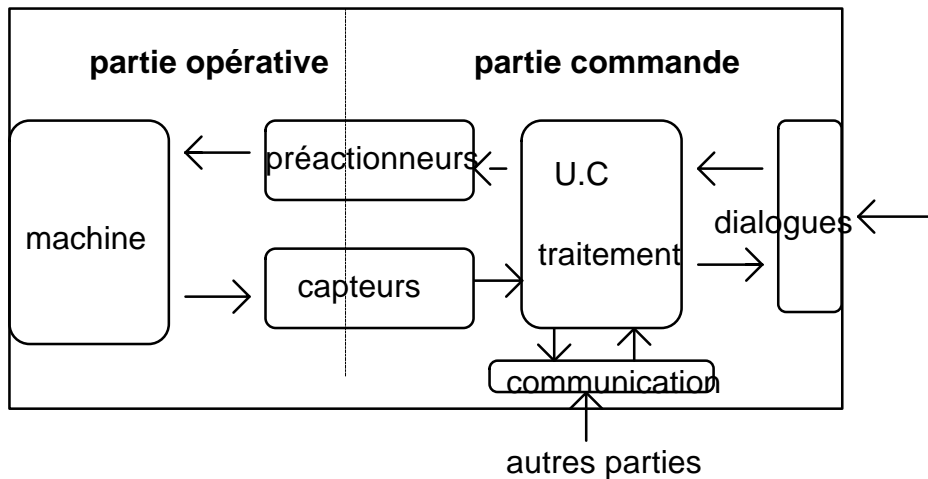
TD11 - Le GRAFCET (outil de description)

I - Système automatisé

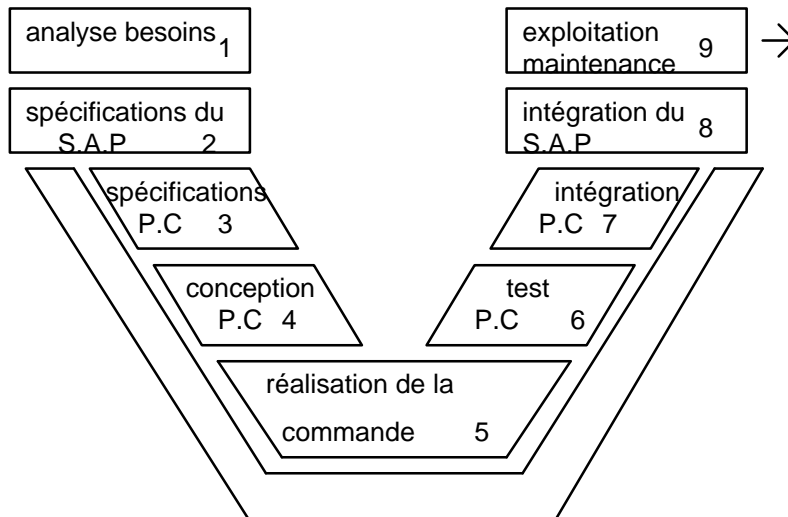
I-1) Généralités.

Un système automatisé se décompose en deux parties dépendantes :

- la **partie opérative** est le processus à automatiser
- la **partie commande** est l'automatisme qui en fonction des entrées (information externe venant de la partie opérative, consignes extérieures, etc.....) élabore en sortie des ordres externes destinés à la partie opérative ou à des éléments extérieurs .



I-2) La démarche de conception d'un système automatisé de production.



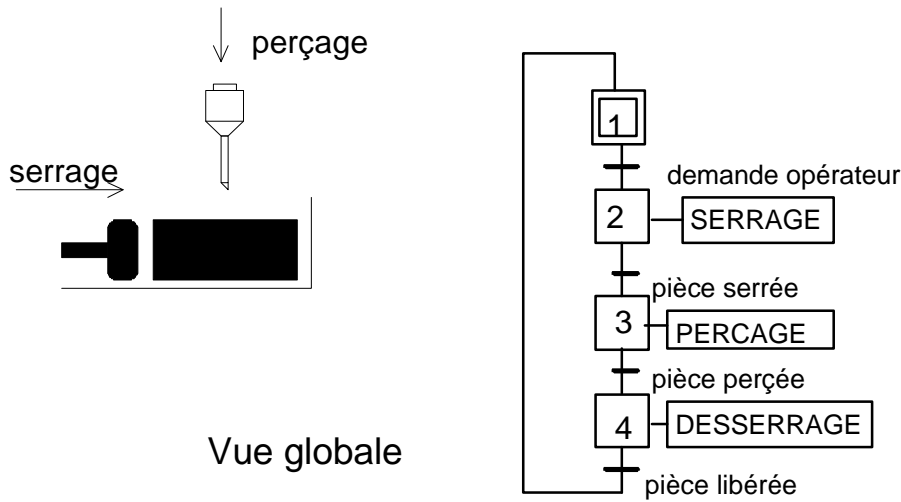
SAP : système automatisé de production.
PC partie commande.

Pendant la démarche de conception, l'information sur l'automatisme est affinée, des outils de représentation (grafcet, gemma,...) sont utilisés permettant la communication entre les différents intervenants impliqués dans le processus depuis la phase de conception jusqu'à l'exploitation.

II - Description par GRAFCET.

Le grafcet permet de représenter par affinements successifs le comportement de la partie commande d'un système automatisé de production selon les souhaits du concepteur.

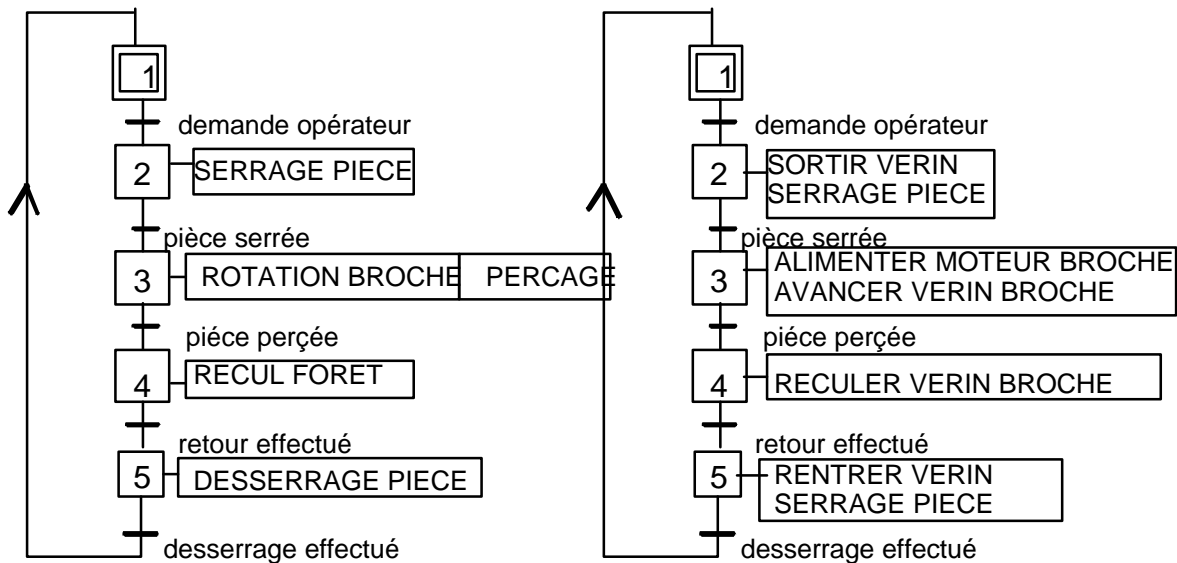
II-1) Exemple : description progressive d'un système automatisé de perçage :



Vue globale

Le niveau global représente le point de vue de l'utilisateur, il décrit les interactions du système sur le produit :

- après avoir effectué la mise en place de la pièce l'opérateur demande le perçage(état 1)
- au reçu de cette information la partie commande donne l'ordre de serrage (état 2)
- après contrôle du bon serrage de la pièce l'ordre de perçage est alors donné (état 3)
- le perçage terminé l'ordre de desserrage est donné (état 4)
- l'automatisme revient en position initiale (état 1)



Vue du concepteur

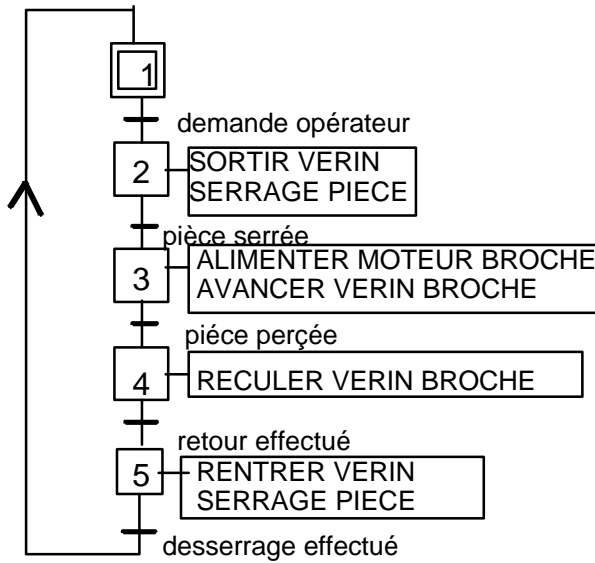
Vu au niveau de la commande

Le concepteur du procédé décrit les différentes actions effectuées par la partie opérative. Le grafcet vu au niveau de la commande concerne l'automaticien. Il décrit tous les ordres que l'équipement de commande doit émettre pour obtenir les effets désirés. C'est ce grafcet que nous retenons pour concevoir le programme de l'automate, le câblage de la commande.

II-2) Différents niveaux de spécification.

Reprenons l'exemple précédent.

Ce grafcet ne présume pas des choix technologiques (capteurs, préactionneurs,...) de la partie opérative ni de la partie commande.



Vu au niveau de la commande

Si nous effectuons les choix techniques suivants:

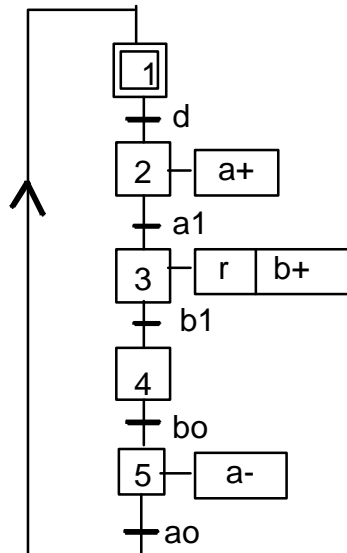
Capteurs :

- demande opérateur : bouton poussoir d
- pièce serrée : capteur fin de course a1
- pièce percée : capteur fin de course b1
- retour effectué : capteur fin de course bo
- desserrage effectué : capteur fin de course ao

Actionneurs :

- mouvement de serrage : vérin double effet commandé par un distributeur 5/2 bistable
a + =serrage
a- =desserrage
- mouvement de descente : vérin simple effet commandé par un distributeur monostable b+ =descente broche
- rotation forêt : moteur électrique commandé par contacteur r

Le grafcet tenant compte des contraintes opérationnelles devient :



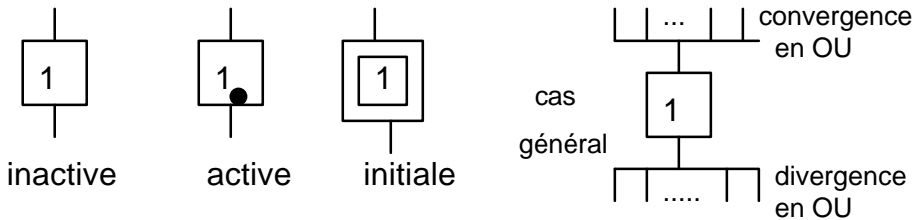
III - Le GRAFCET (règles d'établissement).

III-1) Etapes.

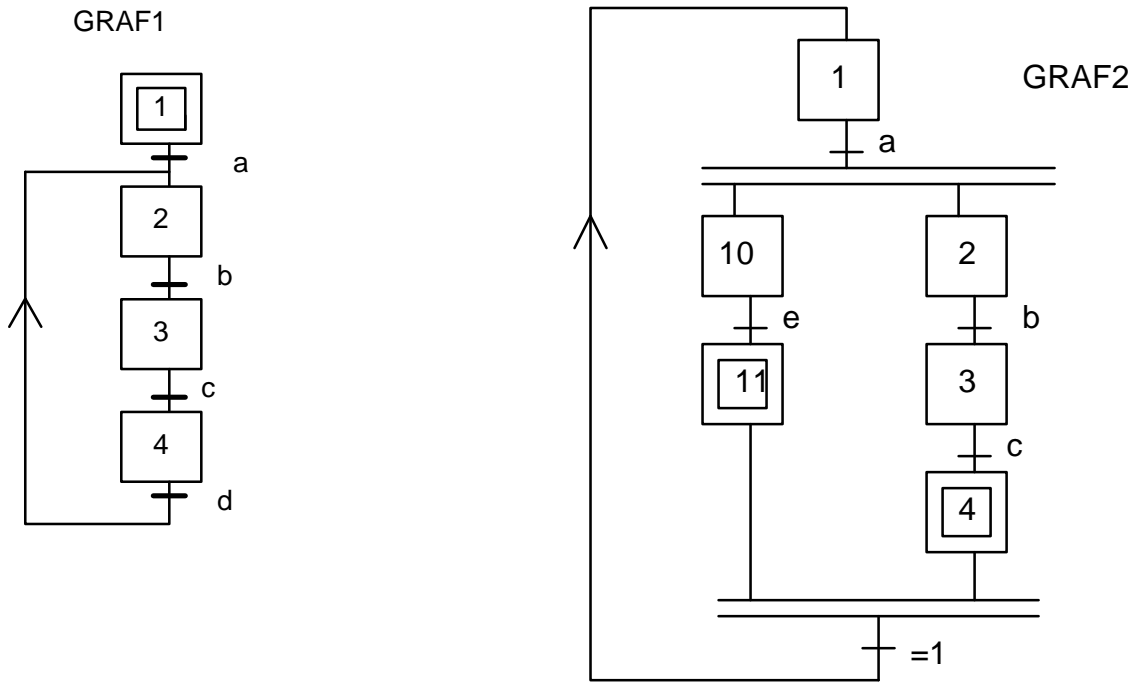
Une étape est représentée par un carré. Elle peut avoir deux états :

- état actif (une marque à l'intérieur du carré)
- état inactif

Règle 1 : La situation initiale d'un grafcet caractérise le comportement de la partie commande vis à vis de la partie opérative et correspond aux étapes actives au début du fonctionnement de la partie commande .

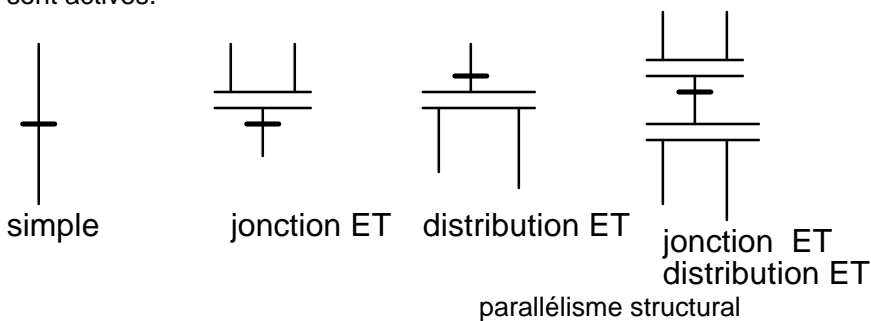


- Remarques :
- la situation initiale peut n'être obtenue qu'une seule fois à la mise sous tension
 - on peut utiliser plusieurs étapes initiales :



III-2) Transitions.

Une transition est représentée par un trait horizontal. A chaque transition est associée une réceptivité. Une transition est validée lorsque toutes les étapes immédiatement précédentes reliées à cette transition sont actives.



Règle 2 : l'évolution de la situation d'un grafcet correspondant au franchissement d'une transition ne peut se faire :

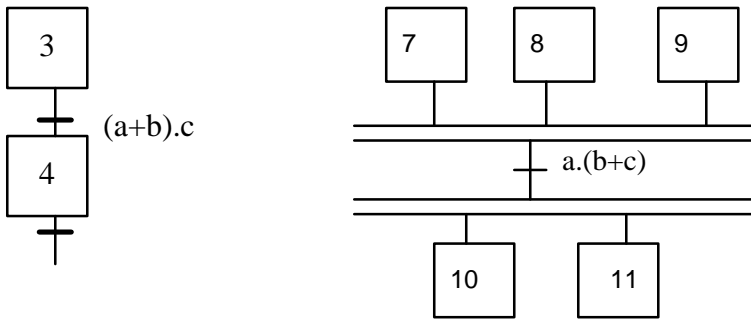
- que lorsque cette transition est validée
- **et que** la réceptivité associée à cette transition est vraie .

Lorsque ces **deux conditions** sont réunies, la transition devient franchissable et elle est obligatoirement franchie.

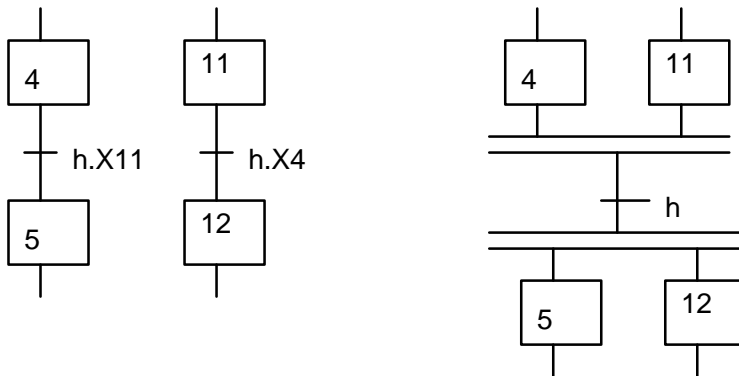
Règle 3 : Le franchissement d'une transition provoque :

- la désactivation de toutes les étapes immédiatement précédentes reliées à cette transition .
- l'activation de toutes les étapes suivantes reliées à cette transition .

exemples :

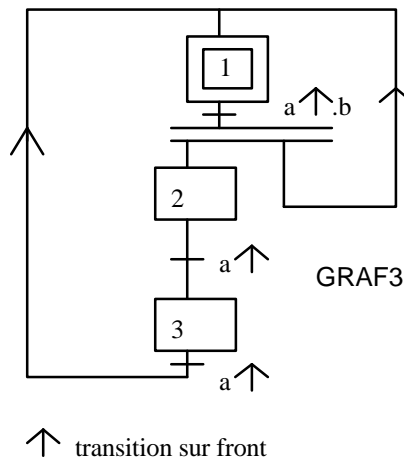


Règle 4 : plusieurs transitions simultanément franchissables sont simultanément franchies
exemple : montrer que ces deux représentations sont équivalentes



Règle 5 : si au cours du fonctionnement de l'automatisme une même étape doit être simultanément activée et désactivée, elle reste active.

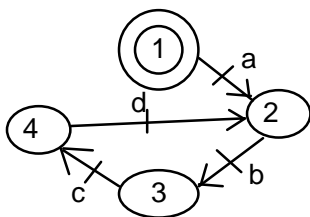
exemple :



III-3) Graphe des états.

Le graphe des états est une représentation "développée" du GRAFCET : c'est le GRAFCET équivalent avec une restriction : il y a toujours un et un seul jeton dans ce GRAFCET. Pour le distinguer du GRAFCET on le représente à l'aide d'étapes rondes (ce sont des états).

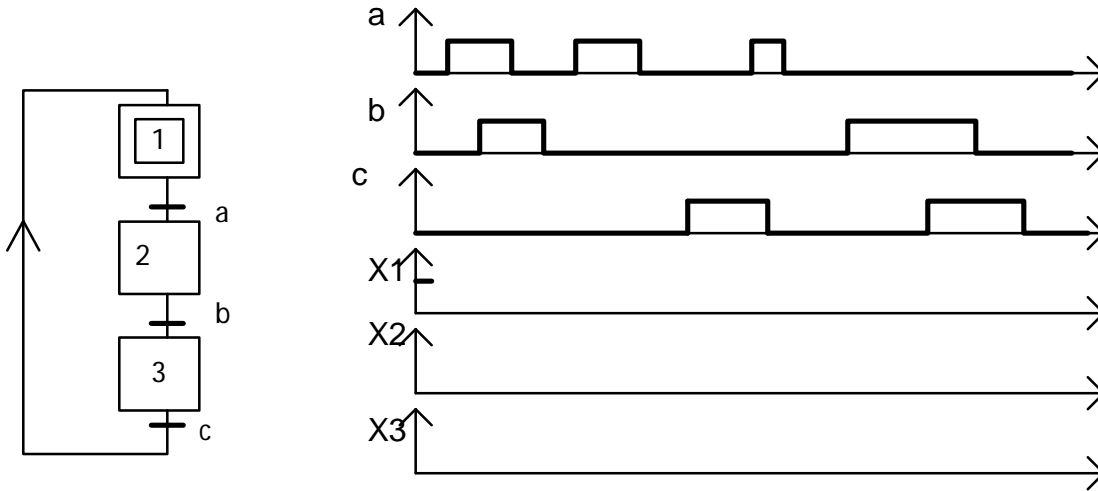
Exemple : on donne ci-dessous le graphe des états du GRAFCET noté GRAF1 du texte.



IV - EXERCICES.

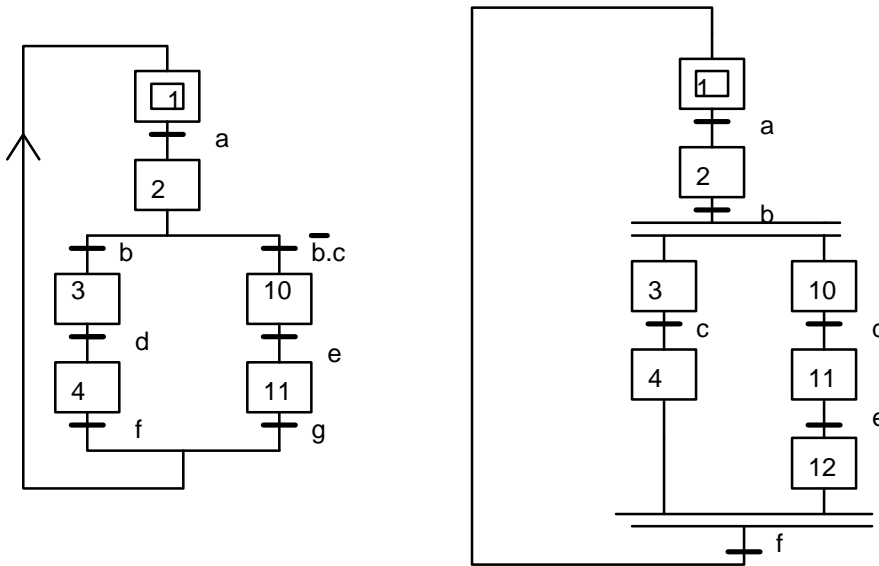
EXERCICE 1 : Etablir le graphe des états des grafcet noté GRAF2 et GRAF3 dans le texte.

EXERCICE 2 : on considère le grafcet suivant, compléter le chronogramme



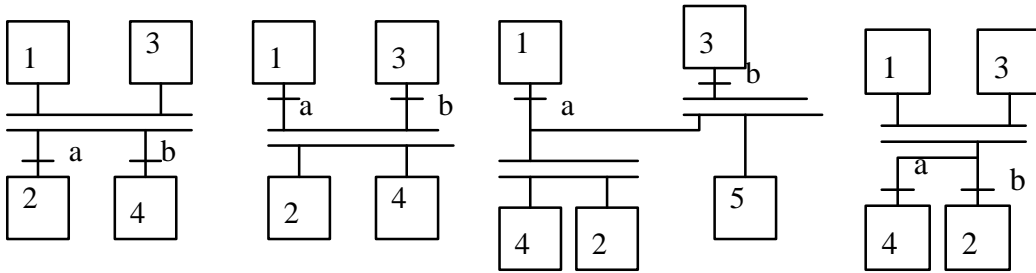
Quel est le nombre d'états possibles ?
Tracer un graphe des états .

EXERCICE 3 : On donne les deux grafcets ci-dessous :



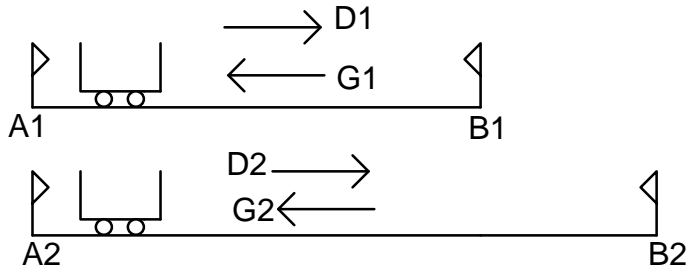
Tracer le graphe des états.
Quel est le nombre d'états possibles dans les deux cas ?
Que se passe-t-il si l'on remplace la transition /b.c par c ? Commenter.
Que se passe-t-il si l'on remplace la convergence en ET par une convergence en OU ?

EXERCICE 4 : respect de la syntaxe

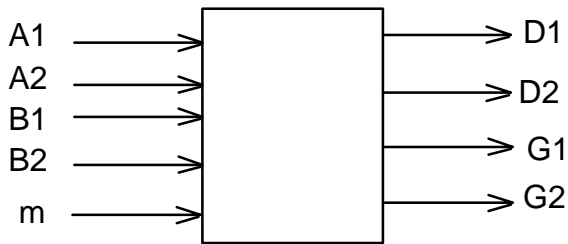


Trouver les erreurs de syntaxe et rétablir la forme correcte .

EXERCICE 5 :

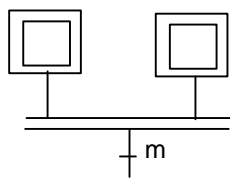


Les deux chariots C1 et C2 sont supposés initialement en position de référence (A1 pour C1, A2 pour C2). Chacun des deux chariots effectue un aller et retour dès réception de la consigne (m). Un nouveau départ n'est possible que si les chariots sont revenus en position de référence.



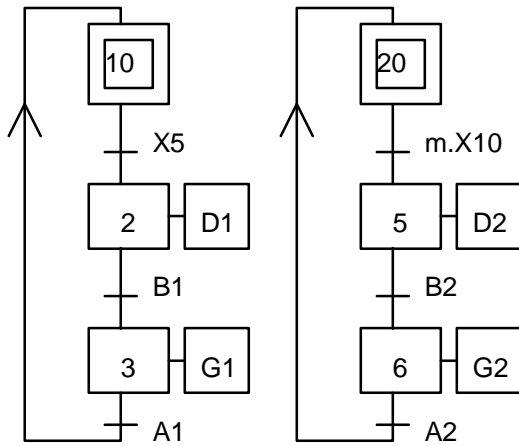
flot des données (schéma fonctionnel)

- 1 Représenter le grafcet en utilisant le parallélisme structural
- 2 Construire une variante de ce grafcet en remarquant que l'on peut transformer les deux étapes de synchronisation en étapes initiales (voir le principe ci-dessous)



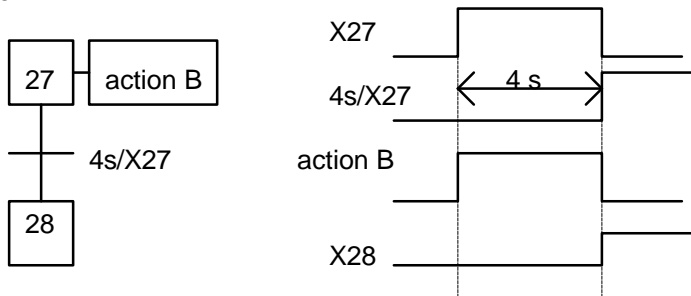
principe

- 3 Construire une autre variante en utilisant deux grafcets séparés.
- 4 Représenter à partir du grafcet de la question 2 le graphe des états.
- 5 Montrer que la solution ci-dessous relève d'une erreur de conception.



EXERCICE 6 : Utilisation de temporisations.

Exemple :



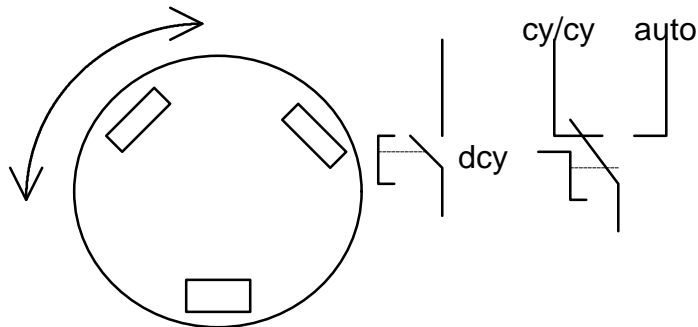
a) On veut faire la sélection d'une impulsion longue ($> 3s$) : l'action sur un bouton poussoir (BP) doit être maintenue pendant un temps minimum de trois secondes pour provoquer une action B.

b) Même exercice pour une impulsion courte ($< 1s$).

EXERCICE 7 : séquences simultanées.

Plateau tournant à 3 postes

- poste 1 : chargement évacuation
- poste 2 : perçage
- poste 3 : taraudage.



Faire le GRAFCET de cette machine à trois postes : les trois opérations doivent se faire en même temps afin de gagner du temps (le GRAFCET restera descriptif, on ne détaillera pas les opérations de perçage, taraudage et chargement-évacuation). Il est demandé de prévoir deux fonctionnements possibles :

- automatique (marche continue sans appui sur dcy)
- cycle par cycle (il faut appuyer sur dcy pour démarrer chaque cycle).

Pour simplifier, on supposera :

Lors de l'arrêt on laissera le plateau dans l'état : sans le vider.

Lors d'un départ on supposera le plateau comme laissé par un arrêt, c'est à dire prêt à recevoir les actions chargement/évacuation, perçage et taraudage.

TD12 - GRAFCET (synthèse matérielle non programmée)

I - Voir le GRAFCET comme une description de machine séquentielle

Les GRAFCETs faisant l'objet d'une synthèse doivent comporter certaines restrictions. Certaines sont obligatoires, d'autres sont conseillées pour simplifier la synthèse :

- pas d'utilisation de variable temporisées (obligatoire),
- non utilisation de transition sur front (conseillée).

La synthèse matérielle d'un GRAFCET est une opération plus ou moins simple suivant la méthode utilisée. Nous allons maintenant en présenter trois. Les circuits matérialisant des GRAFCETs seront appelés des séquenceurs.

II - Généralités

Avant de commencer à réaliser la synthèse, un nombre de concepts important est à définir, ainsi que les manières de les écrire :

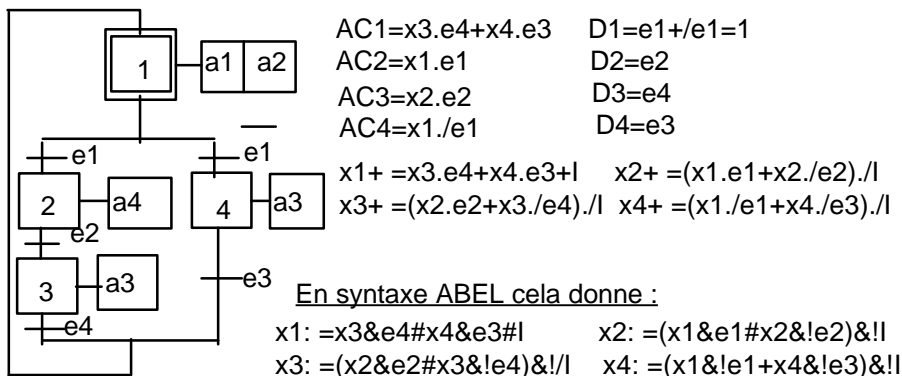
- écrire les conditions d'activations AC_i , une par étape, en fonction des variables d'étapes et des entrées
- écrire les conditions de désactivation D_i , une par étape, en fonction des entrées en considérant que l'on est dans l'étape x_i .

Vient ensuite l'écriture des équations de récurrences : état x_i^+ est fonction de l'état antérieur x_i et des conditions d'activation-désactivation, avec deux possibilités :

- pour une étape initiale : $x_i^+ = AC_i + /D_i \cdot x_i + I$.
- pour une étape normale : $x_i^+ = (AC_i + /D_i \cdot x_i) \cdot I$

I est une entrée spéciale destinée à initialiser le GRAFCET.

Exemple :



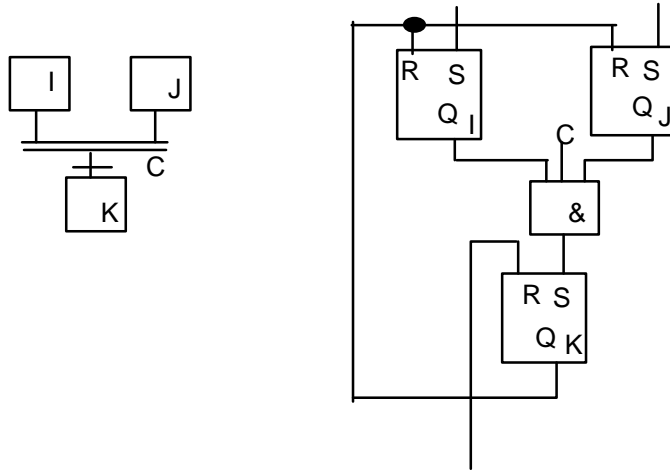
III - Méthode asynchrone

On associe une mémoire RS à chaque étape. La quatrième règle de franchissement impose de prendre des mémoires à S prioritaire. L'état logique des sorties de la bascule reflète alors l'état actif ou non de l'étape correspondante.

Cette méthode consiste à

- élaborer les AC_i à l'aide de circuits combinatoires et à les amener sur le SET des mémoires,
- élaborer les D_i à l'aide de la sortie des mémoires, ceci pour éviter les aléas de fonctionnement, et les amener sur les RESET des mémoires.

Par exemple :



IV - Méthodes synchrones

Il existe au moins deux méthodes synchrones, une simple qu'il est absolument nécessaire de connaître, et une plus compliquée que nous ne ferons que signaler.

IV-1) Méthode d'activation-désactivation synchrone

Cette méthode est à connaître parfaitement.

Elle reprend pour l'essentiel la méthode asynchrone, à savoir une bascule (et non plus une mémoire) par étape. Les bascules utilisées seront évidemment des bascules D.

L'idée générale est d'élaborer pour chaque étape :

$$x_i^+ = AC_i + /D_i \cdot x_i$$

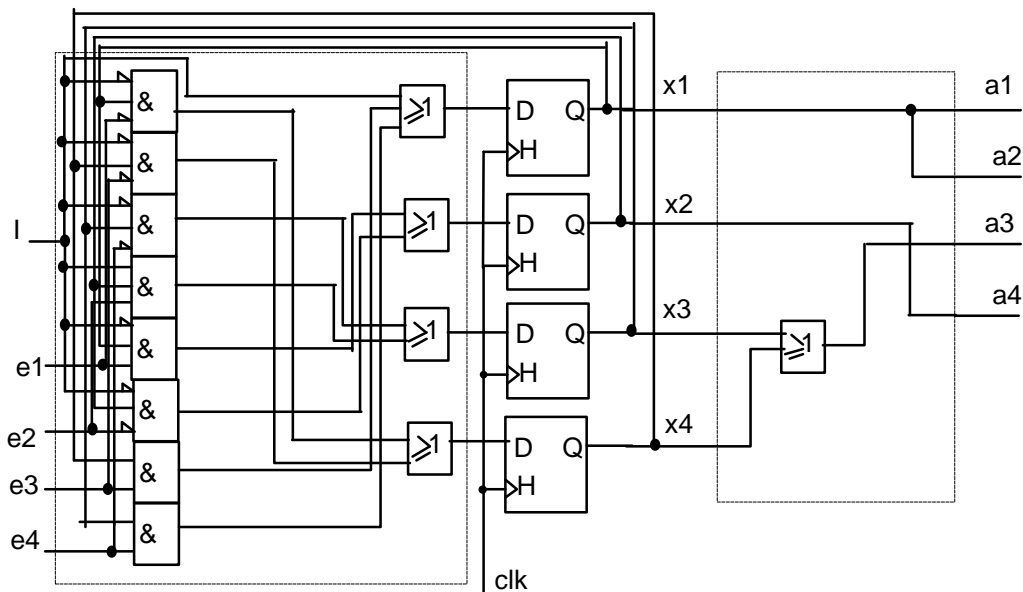
Chaque x_i sera amené sur l'entrée D de la bascule correspondante.

En fait si l'on veut fouiller un peu plus il faut prévoir une entrée I destinée au marquage initial et les relations à implanter par de la circuiterie combinatoire sont :

$$x_i^+ = AC_i + /D_i \cdot x_i + I \text{ pour les étapes initiales,}$$

$$x_i^+ = (AC_i + /D_i \cdot x_i) \cdot /I \text{ pour les étapes ordinaires.}$$

Exemple : (pour le GRAFCET de la page précédente).



Remarques : Cette méthode de synthèse est simple mais pas complètement adaptée aux circuits programmables séquentiels puisqu'elle nécessite beaucoup de bascules (1 par étape) et peu de

combinatoire. Les circuits programmables comportent en général au contraire beaucoup de portes combinatoires et peu de bascules. Ceci n'est plus forcément vrai pour les circuits modernes.

On peut se demander quel est l'intérêt d'ajouter l'entrée d'initialisation I. En fait il faut bien comprendre que dans cette synthèse nous laissons de côté un nombre important d'états. Il faut bien prévoir un moyen de revenir à un état connu si par hasard on se retrouve dans un état non prévu (par exemple 3 jetons dans le GRAFCET ci-dessus !

IV-2) Méthode optimisée en nombre de bascules

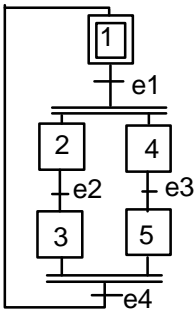
Il est clair que :

- un système de n bascules comporte 2^n états,
- un GRAFCET de m étapes comporte 2^m états au maximum, mais bien souvent beaucoup moins.

On en conclut que souvent $m \ll n$. Il est possible dans une synthèse d'utiliser bien moins de bascules que d'étapes. Pour cela, il suffit de :

- compter les états du GRAFCET et d'en déduire ainsi le nombre de bascules nécessaire,
- coder les états et définir un graphe d'évolution ou graphe d'état,
- implanter ce graphe.

Exemple : Nous allons donner un exemple complet maintenant.



Assignment	
Etats	Codes
1	000
2,4	001
3,4	010
2,5	011
3,5	100

Rappel des équations de récurrences
(ne sont pas utiles pour cette synthèse)

$$\begin{aligned}
 x1+ &= x3.x5.e4+x1./e1+l \\
 x2+ &= (x1.e1+x2./e2)/l \\
 x3+ &= (x2e2+/(x5e4)x3)/l \\
 x4+ &= (x1e1+e3x4)/l \\
 x5+ &= (x4e3+/(x3e4)x5)/l
 \end{aligned}$$

SYNTHESE OPTIMISEE (Bascules D)

Etat initial	Condition	Etat futur
Q2Q1Q0		Q2+Q1+Q0+
000	$\overline{e1}$	001
000	$e1$	000
001	$\overline{e2.e3}$	001
001	$e2.e3$	010
001	$\overline{e3.e2}$	011
001	$e3.e2$	100
010	$\overline{e3}$	100
010	$e3$	010
011	$\overline{e2}$	100
011	$e2$	011
100	$\overline{e4}$	000
100	$e4$	100

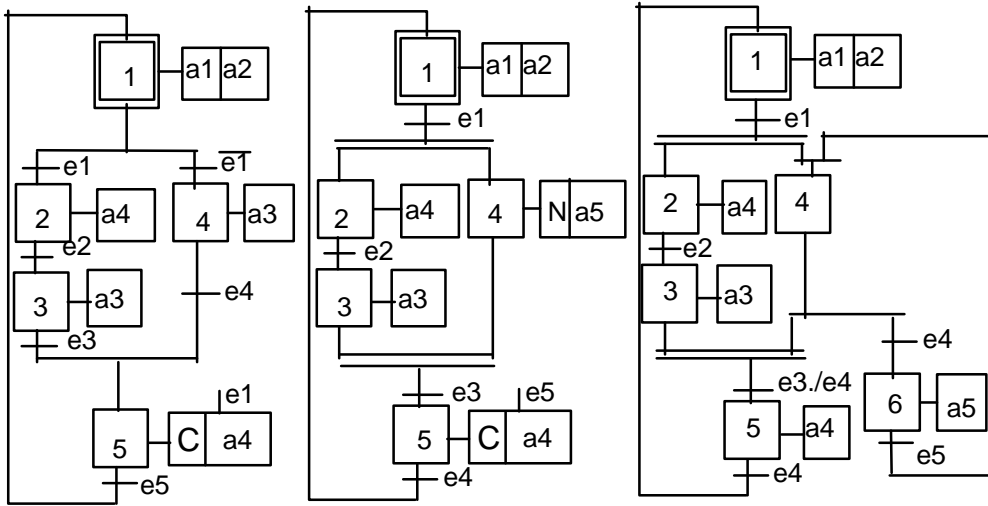
Equations :

$$\begin{aligned}
 Q2+ &= Q0\overline{Q1}\overline{Q2}e3e2+\overline{Q0}Q1\overline{Q2}e3+Q0Q1\overline{Q2}e2+\overline{Q0}Q1\overline{Q2}e4 \\
 Q1+ &= Q0\overline{Q1}\overline{Q2}e2e3+Q0\overline{Q1}\overline{Q2}e3e2+\overline{Q0}Q1\overline{Q2}e3+ \\
 &\quad Q0Q1\overline{Q2}e2 \\
 Q0+ &= \overline{Q0}\overline{Q1}\overline{Q2}e1+Q0\overline{Q1}\overline{Q2}e2e3+Q0\overline{Q1}\overline{Q2}e3e2+Q0\overline{Q1}\overline{Q2}e2
 \end{aligned}$$

La gestion de l'initialisation doublerait la taille du tableau. On peut cependant directement faire intervenir cette initialisation dans les équations de récurrence : &! ici pour toutes les équations car initialisé à $(000)_2$.

V - Exercices

- 1) Réaliser la synthèse des trois GRAFCETs ci-dessous : méthode d'activation désactivation synchrone (une bascule par étape). Remarquer la notation d'une action conditionnelle.
- 2) Réaliser une synthèse optimisée de ces trois GRAFCET.

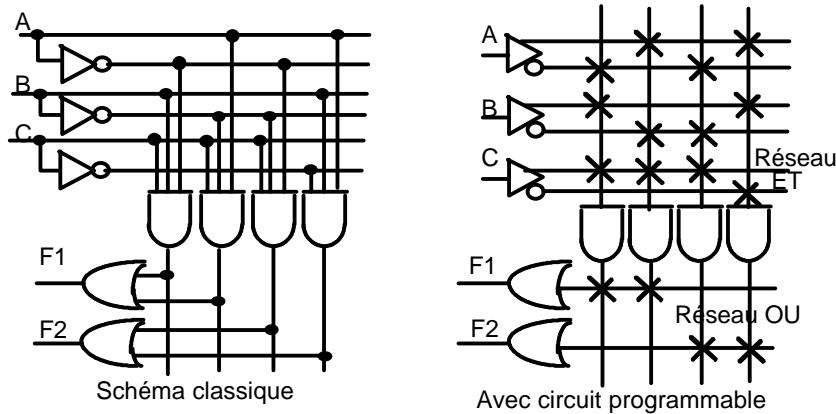


TD 13 - Logique programmable et ABEL

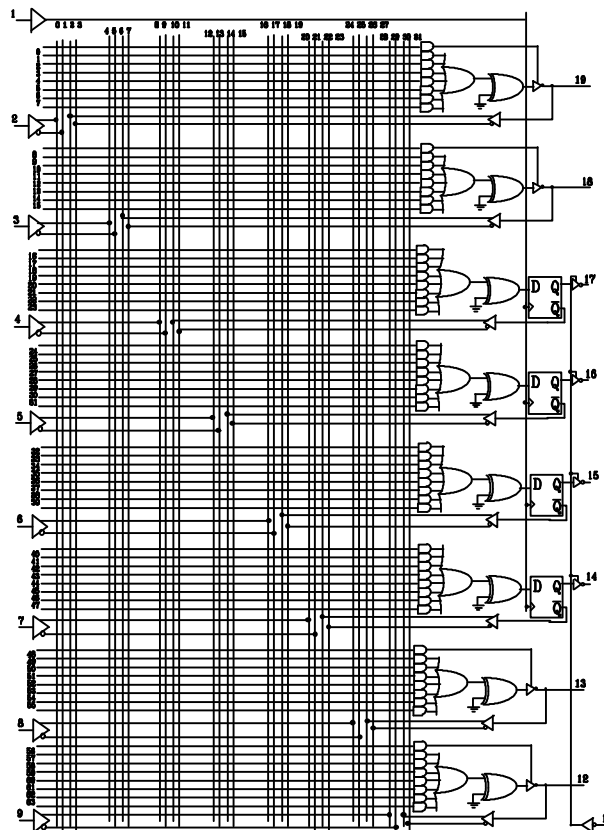
I - Présentation de la logique programmable

I-1) Les conventions de représentation des PALs

Nous vous proposons ci-dessous un schéma logique de deux fonctions F1 et F2 dans les deux conventions, habituelle et propre aux PALs.



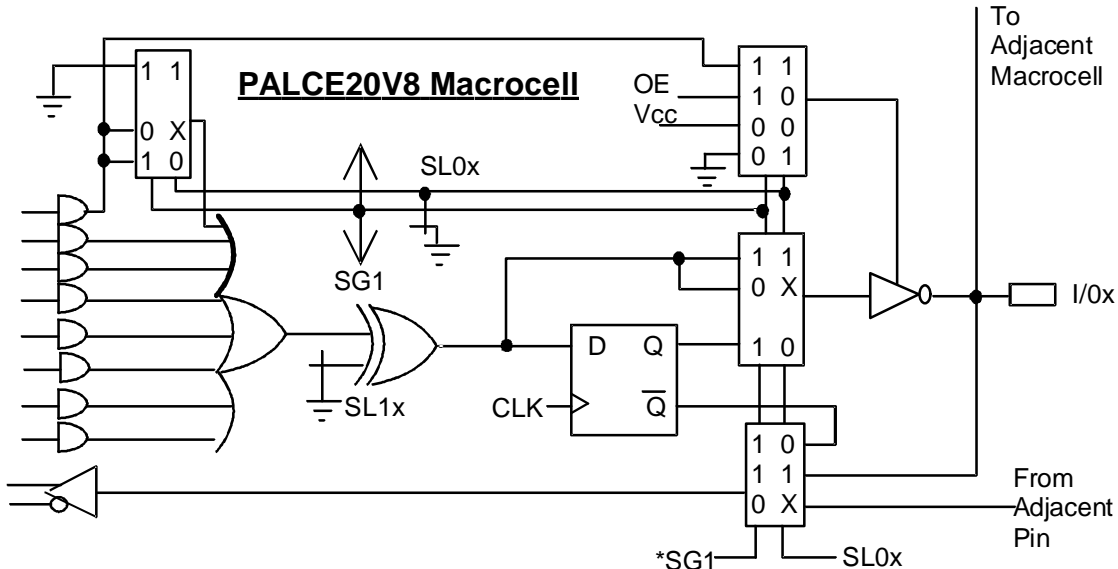
Nous donnons à titre d'exemple le contenu d'une PAL 16 R4 ci-dessous. Le R désigne registered. 4 veut donc dire qu'il y a 4 sorties séquentielles (sur bascules D). Le 16 veut dire qu'il y a possibilité de réaliser 16 entrée.



PLA 16 R4

I-2) Notre GAL 20V8

Nous allons nous intéresser maintenant à un autre type de PAL les GAL de série V (pour versatile) qui veut dire que les sorties peuvent être programmées. Ce qui caractérise ce genre de composant est la notion de macro-cellule comme montrée sur la figure ci-dessous. Une telle cellule est configurable à l'aide de bits internes locaux (appelés SL_n) et globaux (appelés SG_n) où n désigne le numéro du bit et x le numéro de la macro-cellule. Elle est constituée d'autre part par des multiplexeurs qui peuvent permettre plusieurs configurations de la macro-cellule ces derniers étant déterminés par les 2 bits SG1 et SL0_x.



*In macrocell MC0 and MC7, SG1 is replaced by /SG0 on the feedback multiplexer.

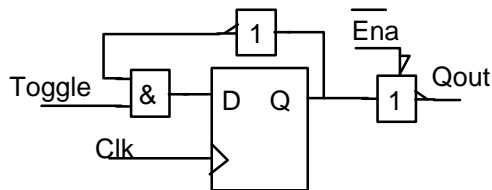
II - Exercices

1°) Trouver à l'aide du schéma de la macro-cellule de la GAL 20V8 tous les schémas équivalents possibles. Donner la liste des configurations impossibles pour 2 macro-cellules distinctes (bits globaux incompatibles).

2°) On donne un programme ainsi que le schéma équivalent associé.

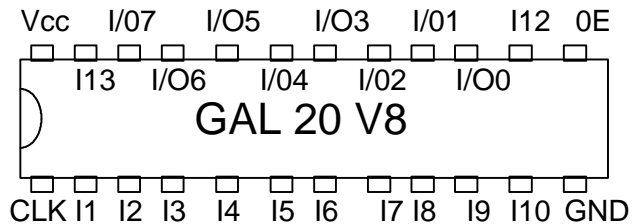
```

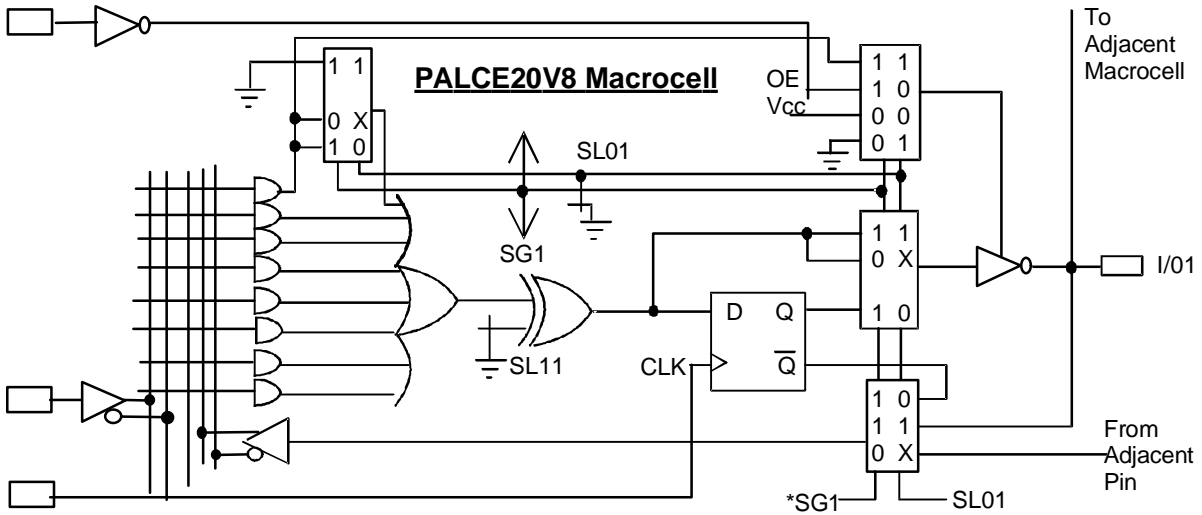
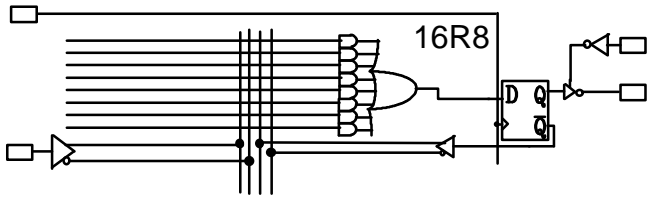
module exo2
  Clk pin 1;
  Toggle pin 2;
  !Ena pin 11;
  Qout pin 16 istype 'reg';
equations
  !Qout := Qout.FB & Toggle;
  Qout.CLK = Clk;
  Qout.OE = Ena;
test_vectors([Clk,!Ena,Toggle] -> Qout)
  [.c.,0,0]->.X.;
  [.c.,0,1]->.X.;
  [.c.,0,1]->.X.;
  [.c.,0,1]->.X.;
  [.c.,0,1]->.X.;
  [.c.,1,1]->.X.;
  [.c.,0,1]->.X.;
  [.c.,1,1]->.X.;
  [.c.,0,1]->.X.;
end
  
```



a) Donner l'ensemble des résultats des vecteurs tests présentés dans le programme. (Sorties possibles 0, 1, Z)

b) Complétez les schémas ci-dessous : numéro des broches et fusibles non grillés pour les deux architectures 16 R8 et 20v8. Dans ce dernier cas, on vous demande de corriger le programme pour la déclaration des broches et de réfléchir d'abord à la valeur des bits internes.

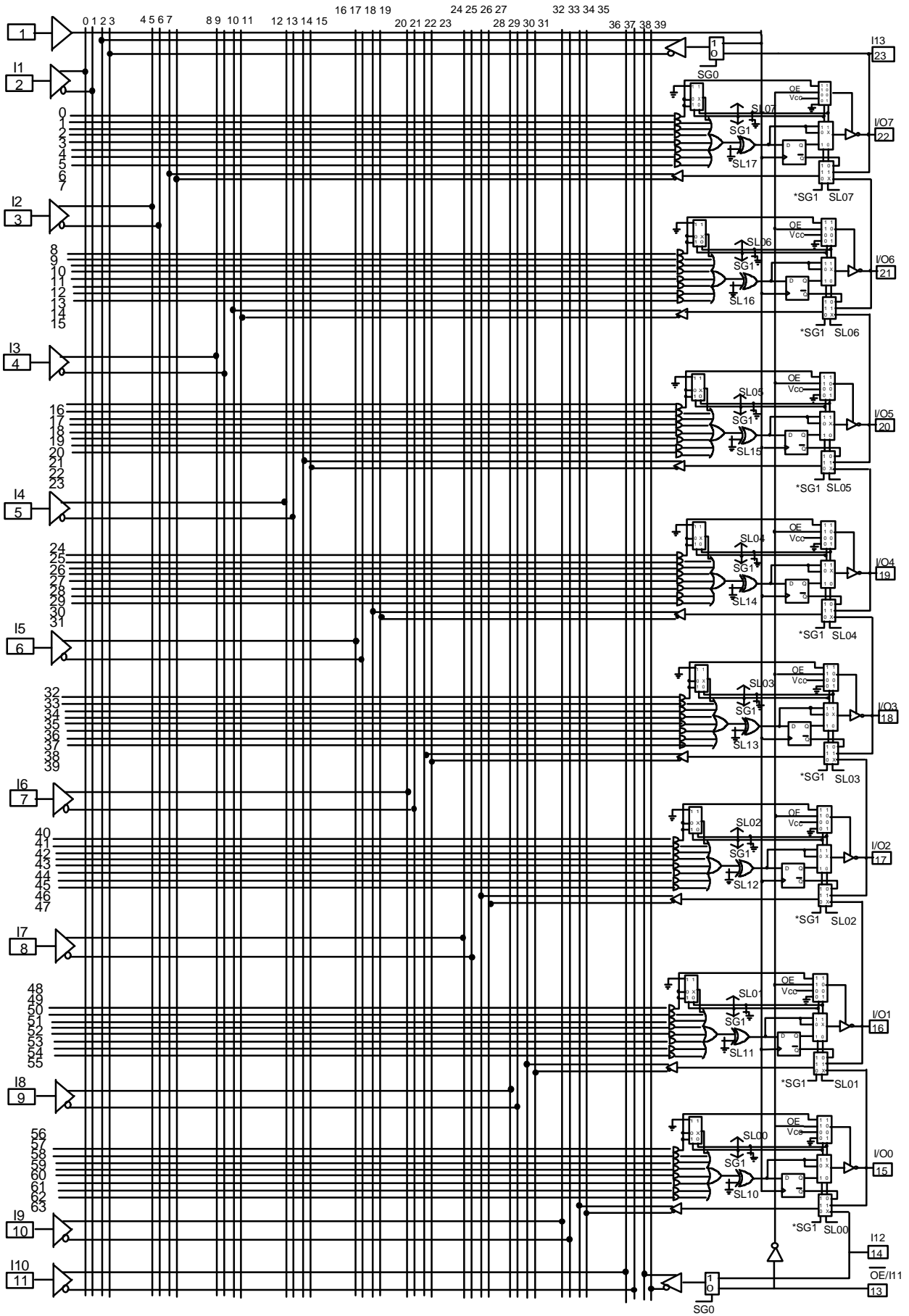


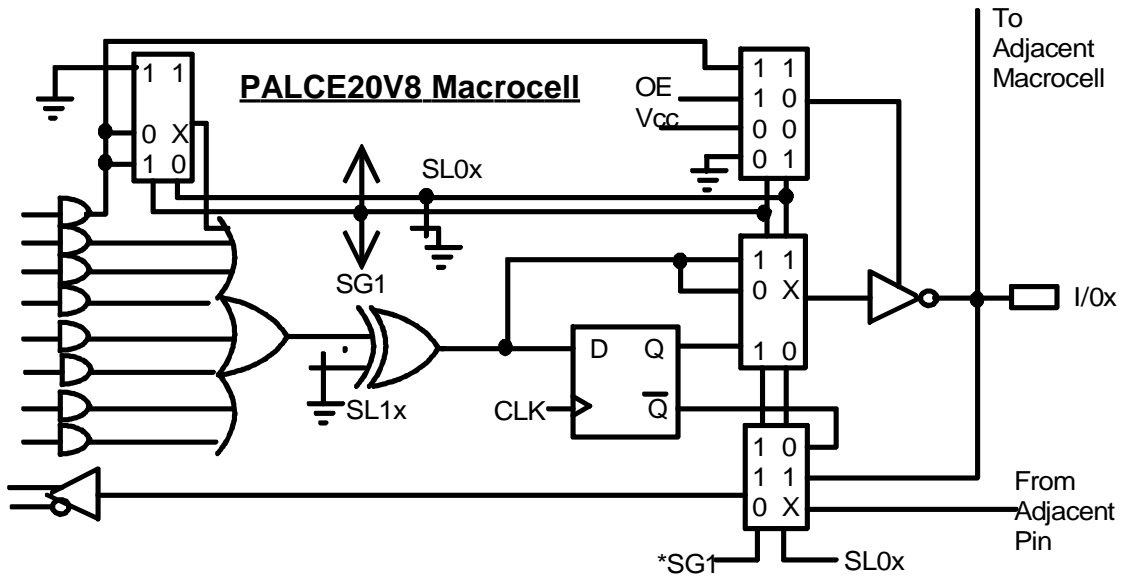


*In macrocell MC0 and MC7, SG1 is replaced by /SG0 on the feedback multiplexer.

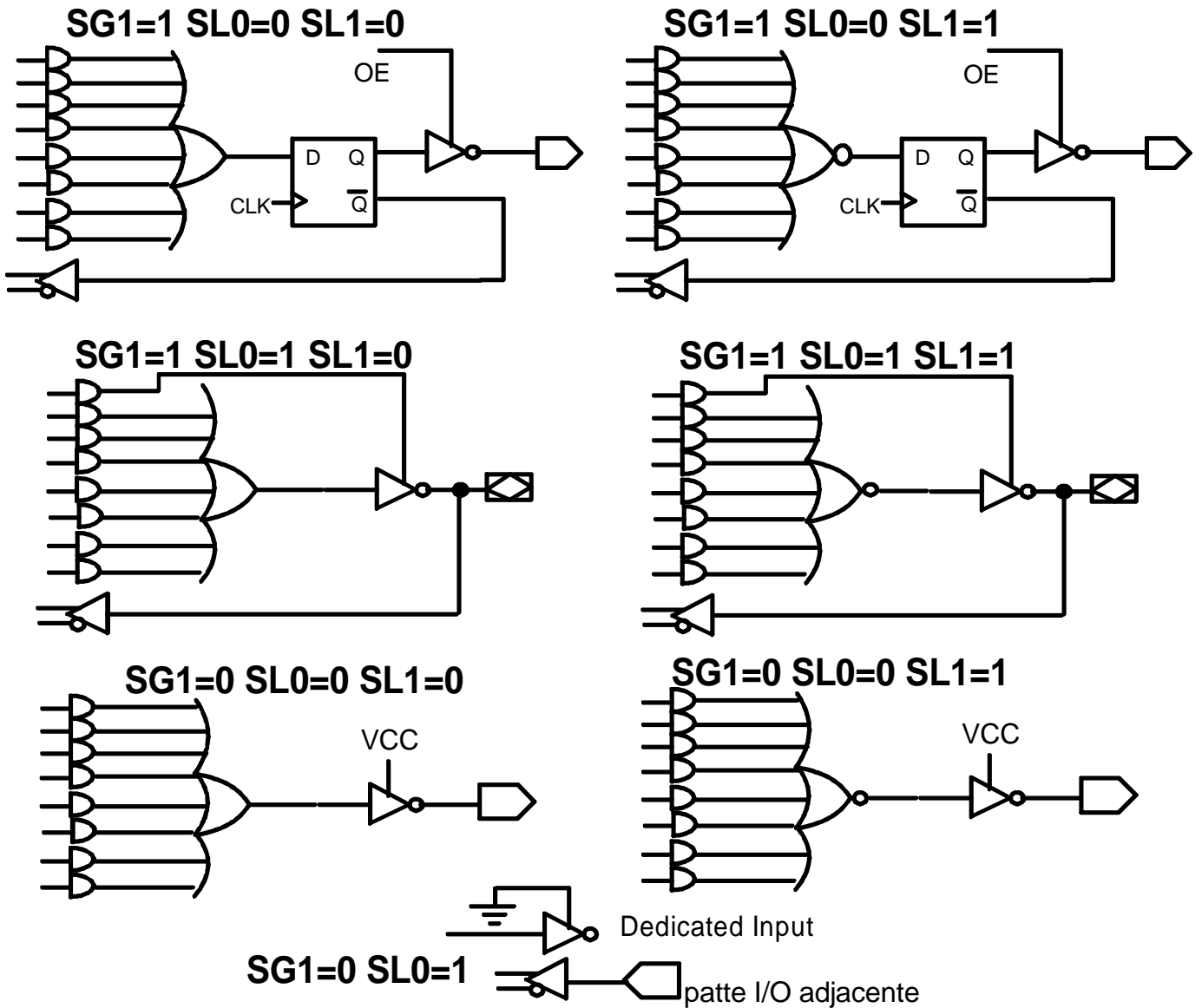
Remarque : Il est possible d'utiliser une sortie en entrée (pattes I/Oi) mais il faut absolument écrire une équation qui oblige la sortie à être en état haute impédance.

Out.oe=0; "par exemple

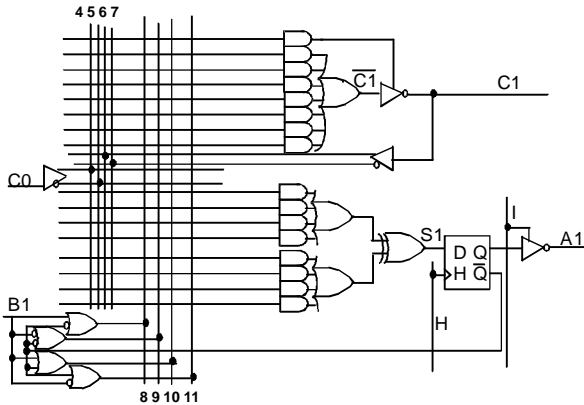




*In macrocell MC0 and MC7, SG1 is replaced by /SG0 on the feedback multiplexer.



ETUDE D'UN PAL 16X4 (spécialisé pour opération arithmétique)



1°) La cellule de base d'un PAL destiné à l'arithmétique est donnée ci-contre. Exprimer les valeurs logiques des fils internes 8, 9, 10 et 11 (voir schéma) en fonction de A1 et B1 :

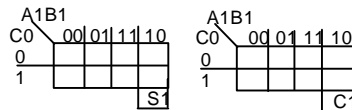
8 =
9 =
10 =
11 =

A quoi sert I ?
Quels fils internes utiliser pour réaliser un ou exclusif ?
..... A1..... ? B1..... ?

2°) Le PAL 16X4 est en fait formé par 4 cellules comme ci-contre. On désire montrer qu'une telle cellule permet d'implanter sur ce PAL un additionneur binaire :

$(C1\ S1)_2 = (A1)_2 + (B1)_2 + (C0)_2$

Remplir les tableaux de Karnaugh ci-dessous



En déduire les équations : S1 =
/C1 =

S1 sera exprimé à l'aide de OU exclusifs, tandis que /C1 sera exprimé sous forme canonique simplifiée. Dessiner les fusibles non grillés sur la figure ci-dessus. On rappelle : $A/B = (A+B)/(A+B)(A+B)$

A quoi servent les bascules D dans l'opération arithmétique réalisée par le PAL ?

IV - COMPLEMENT : PROGRAMMATION D'UN GRAFCET EN ABEL

Nous donnons maintenant un exemple complet de GRAFCET exprimé dans le langage ABEL. Nous utilisons ici des macros ce qui n'est pas obligatoire mais rend le programme plus lisible parce que plus proche des équations de récurrences déjà rencontrées.

```

MODULE EXO12
title 'exercice 12'
exo12 device 'P20V8';
  clock,e1,e2,e3,e4,I pin 1,2,3,4,5,6;
"entrees
  x1,x2,x3,x4,x5 pin 15,16,17,18,19 istype
'reg';

"sorties sequentielle
  x6 pin 20 istype 'com'; "sortie
combinatoire
AC1 macro {x5&e2#x3&e3}; "activation étape 1
AC2 macro {x1&!e1&e4}; "activation étape 2
AC3 macro {x2&e2}; "activation étape 3
AC4 macro {x1&e1}; "activation étape 4
AC5 macro {x4&e3}; "activation étape 5
D1 macro {e1#!e1&e4}; "desactivation

"etape1
D2 macro {e2}; "desactivation etape 2
D3 macro {e3}; "desactivation etape 3
D4 macro {e3}; "desactivation etape 4
D5 macro {e2}; "desactivation etape 5
equations
[x1,x2,x3,x4,x5].clk = clock; "obligatoire
x1 := AC1 # !(D1) & x1 # I; "etape initiale
x2 := AC2 & !I # !(D2) & x2 & !I; "etape 2
x3 := AC3 & !I # !(D3) & x3 & !I; "etape 3
x4 := AC4 & !I # !(D4) & x4 & !I; "etape 4
x5 := AC5 & !I # !(D5) & x5 & !I; "etape 5
x6 = x5 # x2; "equation combinatoire
end
                    
```

TD 14 - GRAFCET : Synthèse programmée.

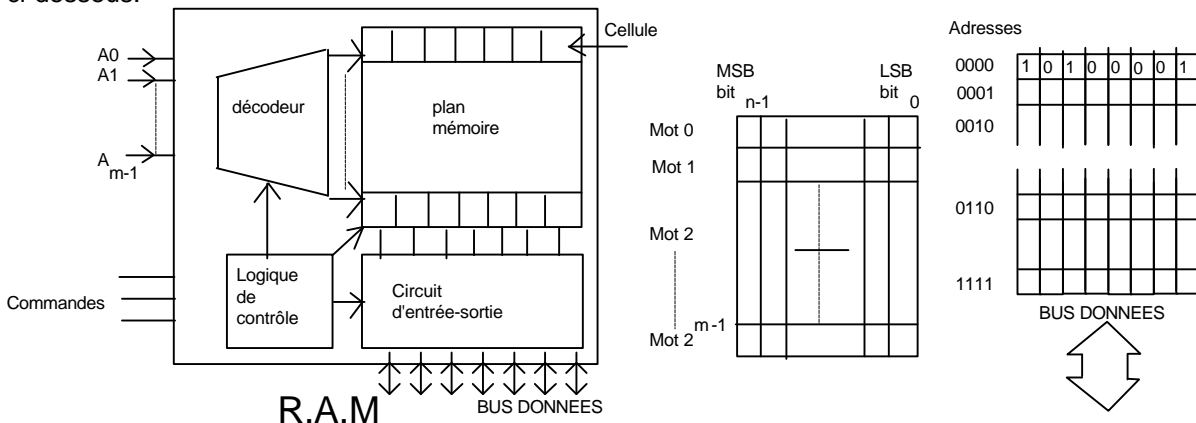
Nous allons aborder maintenant l'implantation programmée d'un GRAFCET. Cette implantation peut se faire sur des architectures simples (mémoire ROM + registre) comme sur des architectures plus complexes (automate programmable). Affinons la terminologie : une architecture sera qualifiée de simple si elle fait appel au plus à un registre ou un compteur. Le but de ce TD est d'étudier l'implantation sur architectures simples. L'implantation sur automate programmable sera étudiée en TP.

I - Bien comprendre l'architecture.

Avant de commencer de résoudre un problème de synthèse, il est naturellement important de bien comprendre le fonctionnement de l'architecture. Il nous faut d'abord rappeler le fonctionnement d'une mémoire.

I-1) Mémoires.

Il existe plusieurs sortes de mémoires : RAM, ROM, EPROM.... Nous en donnons un schéma fonctionnel ci-dessous.



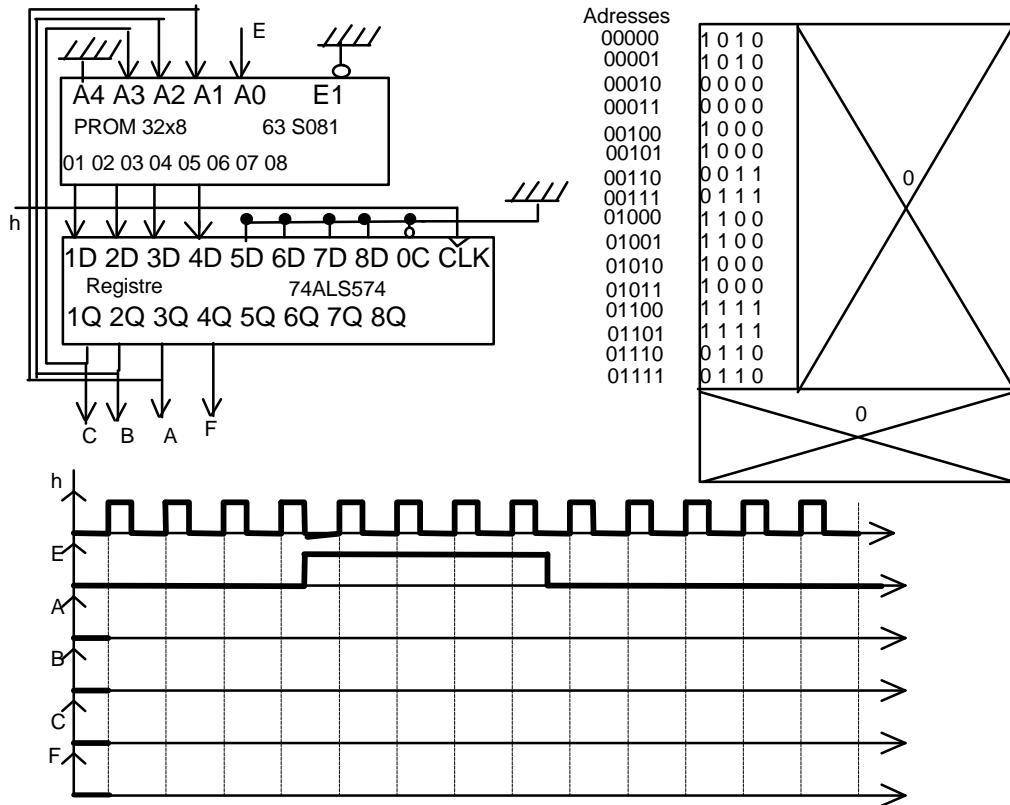
Exercice : Quelle est la largeur du bus de données, du bus d'adresse. On positionne le bus d'adresse à 0000, quelle valeur retrouvera-t-on sur le bus de données lors d'une lecture. On positionne l'adresse à 0110 en écriture avec 00001111 sur le bus de données. Que se passe-t-il ?

Chaque cellule a un numéro appelé adresse. n cellules donc adresses de 0 à n-1. Chaque cellule contient k bits (donc 2^k valeurs possibles). Les adresses contiennent m bits donc 2^m cellules directement adressables. Le nombre de bits d'adresse ne dépend que du nombre de cellules adressables et non de leur taille.

I-2) Analyse.

Nous allons commencer par un travail d'analyse sur un schéma, en vue d'une compréhension de l'architecture.

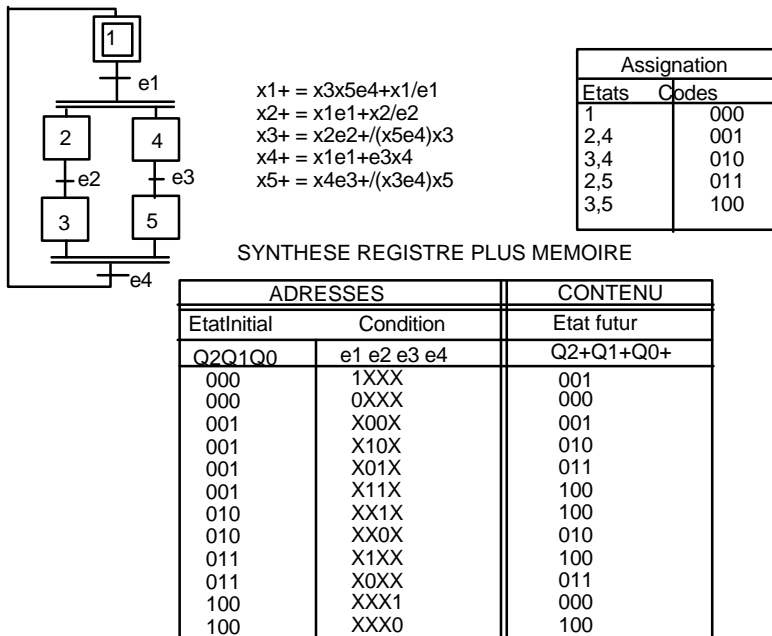
Pour le schéma ci-dessous, dessiner les chronogrammes des sorties en fonction des entrées. Essayer de faire le diagramme de transition pour ce schéma.



II - Synthèse (mémoire plus registre).

La synthèse consiste à partir d'un problème brut, à trouver le GRAFCET puis à réaliser une architecture matérielle et enfin à remplir la mémoire.

Nous allons commencer par donner un exemple complet avec optimisation :



Passons maintenant aux exercices.

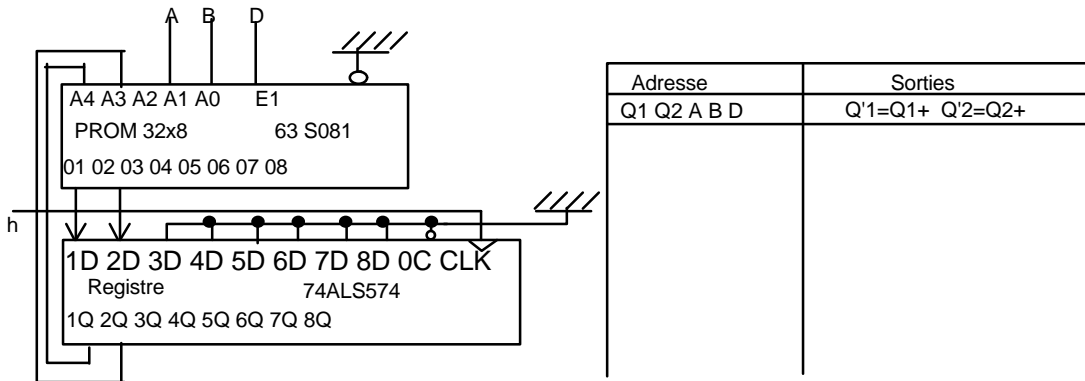
1°) On veut réaliser l'automatisme simplifié suivant :

Un chariot est à l'arrêt en A. Si l'on appuie sur le bouton D de départ, le chariot se déplace en marche avant jusqu'au point B et revient en marche arrière. Il s'arrête lorsqu'il arrive en A. Le cycle peut alors recommencer.

1-a) Décrire cet automatisme à l'aide d'un GRAFCET.

1-b) On donne l'architecture ci-dessous, en déduire le contenu de la mémoire. Pour cela on remplira le tableau donné en commençant par coder chaque étape. Puis pour chaque étape il faut trouver l'état futur dans les deux cas : évolution ou pas.

1-c) Remarquer que ce tableau permet une synthèse rapide en remplaçant la mémoire par un circuit combinatoire (et éventuellement le registre par deux bascules D) : quelles sont les équations à implanter?



III - Synthèse compteur plus mémoire.

La synthèse précédente présente l'inconvénient de nécessiter des mémoires trop importantes. Nous allons voir que l'utilisation d'un compteur et d'une mémoire permet de supprimer ce problème.

Notre nouvelle architecture est constituée au minimum de deux parties :

- une mémoire contenant **les instructions** proprement dites.
- un compteur programme pouvant :
 - * rester dans une même position en attente d'un événement extérieur,
 - * s'incrémenter pour passer à l'instruction suivante,
 - * s'initialiser à zéro,
 - * modifier son contenu (fonction LOAD) pour permettre un saut dans un programme

Format d'une instruction :

- champ opération : ensemble d'éléments binaires qui viennent commander la fonction du compteur
- champ adresse : adresse de condition + adresse de branchement,
- champ action : sorties...

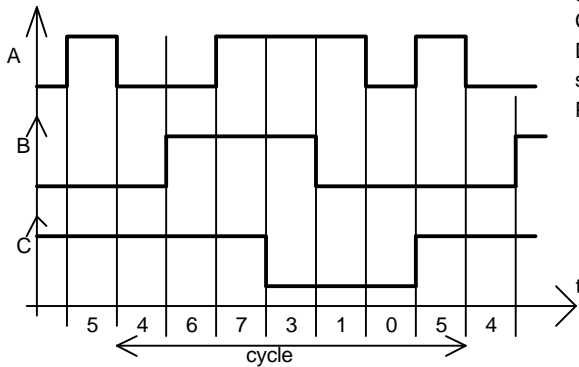
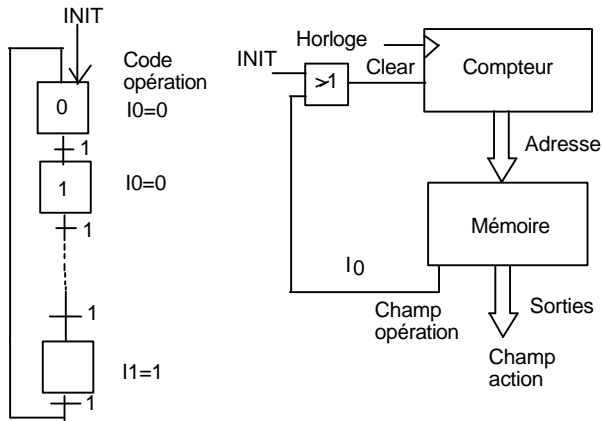
III-1) Séquenceur à enchaînement séquentiel

C'est le système le plus simple, constitué d'un compteur et d'une mémoire programme. Le champ opération ne contient qu'un seul bit, le reste étant pour les sorties. Si ce bit est à 0, on a un enchaînement séquentiel, s'il est à 1 c'est une remise à zéro (RAZ).

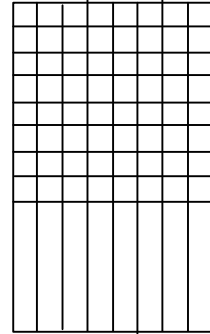
Le langage associé à cette architecture peut être résumé par le tableau :

Bit IO	Instruction
0	NEXT-ST sortie
1	END-ST sortie

Remarque : ce langage est simple mais sa sémantique manque de clarté à cause du fait que chaque instruction réalise en fait deux opérations X et ST !



On désire réaliser un générateur des signaux A,B et C représentés ci-contre. Donner le contenu de la PROM sachant qu'il s'agit d'une PROM 32x8 (63S081).



III-2) Séquenceurs à enchaînement conditionnel

Dans le séquenceur précédent, il n'y a aucune condition logique autorisant le passage d'une étape à une autre. Or la plupart des séquenceurs ne doivent quitter une étape que si la condition de franchissement est vérifiée.

=> code opération :

1er bit comme plus haut,

2eme bit permettant de choisir entre l'enchaînement conditionnel ou séquentiel,

champ adresse de condition (lié au multiplexeur)

Le langage associé à cette architecture peut être résumé par le tableau :

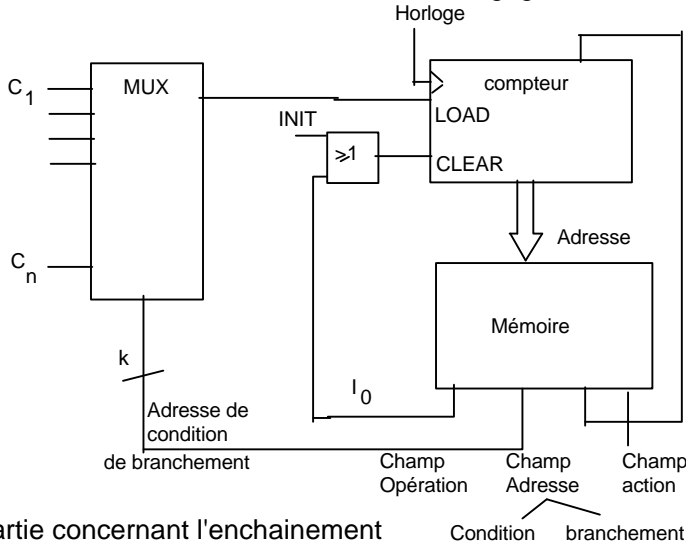
Bit I0	Bit I1	Instruction
0	1	NEXT-ST sortie
1	X	END-ST sortie
0	0	NEXT-IF condition ST sortie

Remarque : ce langage est simple mais sa sémantique manque de clarté à cause du fait que chaque instruction réalise en fait deux opérations X et ST. De plus ici la dernière instruction possède deux opérands. Ce que l'on recherche c'est une architecture pouvant supporter un langage avec un opérateur et 0 ou 1 opérande au plus.

Pour présenter un intérêt, il faut en général l'associer à un séquenceur avec branchement conditionnel.

III-4) Séquenceur avec branchement conditionnel

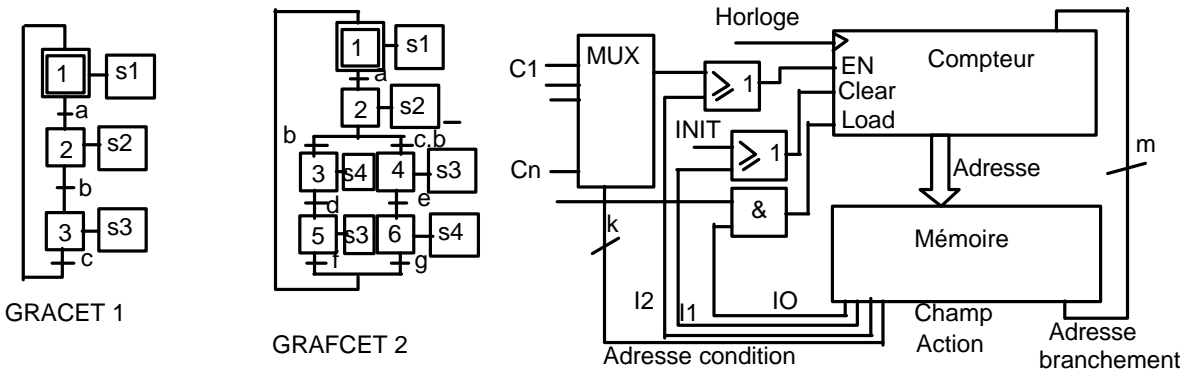
L'objectif est de pouvoir prendre des décisions en fonction de l'état du système commandé. Il correspond à la structure de contrôle de choix dans les langages évolués : le **IF .. THEN .. ELSE** du pascal.



La partie concernant l'enchaînement n'est pas représentée

IV - Exercice

Pour chacun des GRAFCETs présentés, dimensionner le multiplexeur et la mémoire, puis remplir cette dernière à l'aide d'un langage que l'on définira.



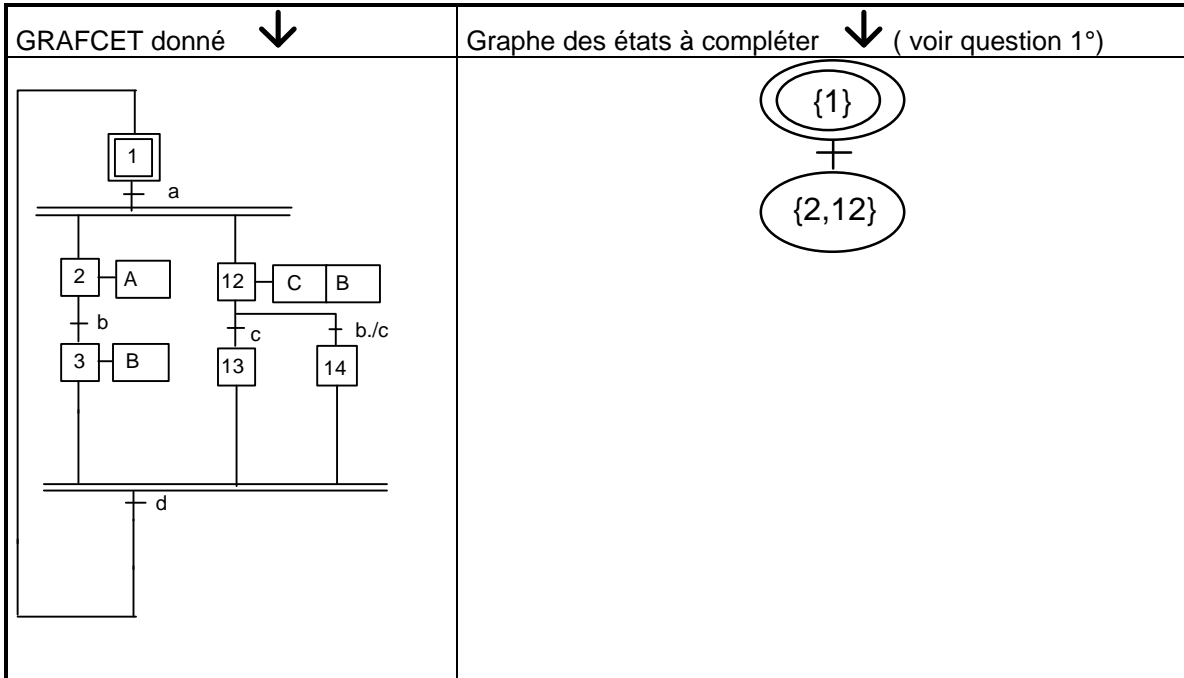
NOM :
Prénom :
Groupe :

Avril 1996

DS Informatique Industrielle n°3 (extraits)

PROBLEME II

On donne le GRAFCET ci-dessous.



1°) La notation {2,12} désigne l'état pour lequel l'étape 2 et l'étape 12 sont actives en même temps. Compléter le graphe des états en page précédente.

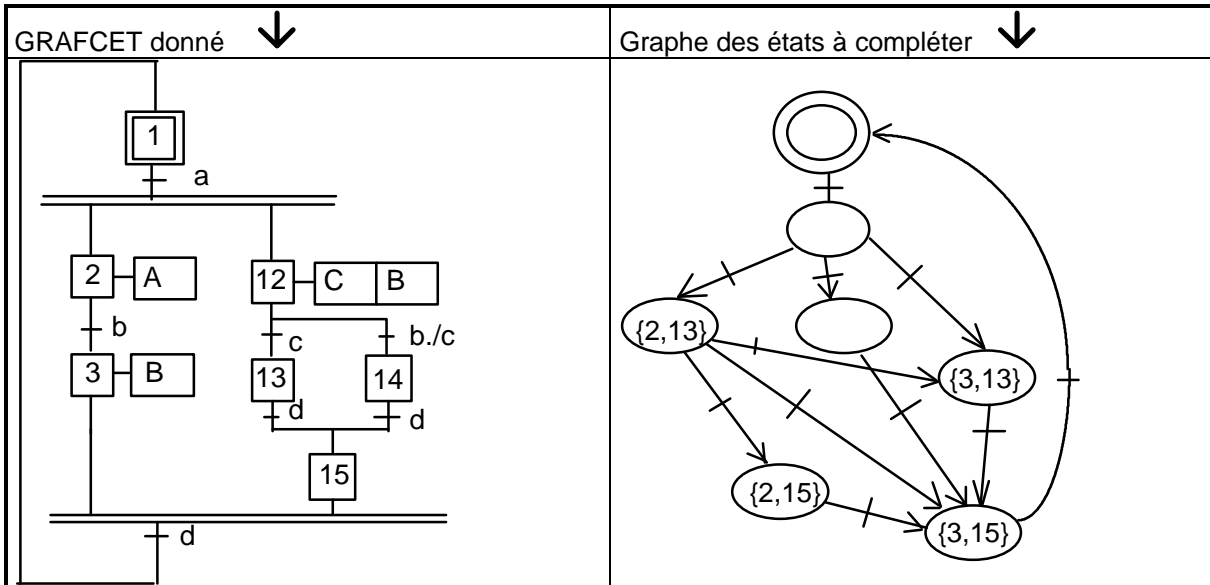
2°) Combien d'états comporte ce GRAFCET ?

Réponse :

3°) Conclusion sur ce grafcet.

Réponse :

4°) Compléter le graphe des états pour le nouveau grafcet corrigé ci-dessous :

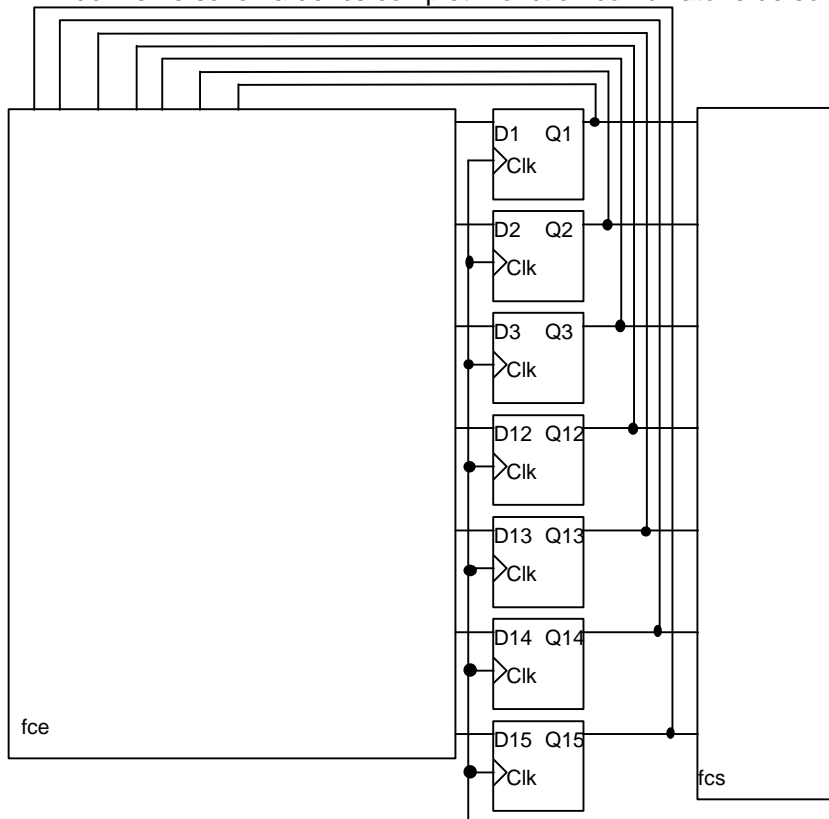


5°) Ecrire les équations de récurrence correspondant au nouveau GRAFCET que l'on vient de compléter. On utilisera une entrée I d'initialisation.

Réponse :

6°) On désire réaliser ce grafcet en utilisant une bascule D par étape (synthèse non optimisée) :

- définir les entrées de fce : fonction combinatoire d'entrée.
- faire le schéma correspondant à l'équation de récurrence de x2 seulement.
- donner le schéma de fcs complet : fonction combinatoire de sortie.



PROBLEME III : implantation de GRAFCET sur GAL 20V8

1°) Recherche de GRAFCET.

On donne le programme ABEL ↓	Dessiner le GRAFCET correspondant ↓
<pre> MODULE EXO3 exo3 device 'P20V8'; clock,e1,e2,e3,l pin 1,2,3,4,5; "entrees x1,x2,x3 pin 19,20,21 istype 'reg'; act pin 22 istype 'com'; "sortie combinatoire AC1 macro {x3&e3};"activation étape 1 AC2 macro {x1&e1};"activation étape 2 AC3 macro {x2&e2};"activation étape 3 D1 macro {e1}; "desactivation etape 1 D2 macro {e2}; "desactivation etape 2 D3 macro {e3}; "desactivation etape 3 equations [x1,x2,x3].clk = clock; "obligatoire x1 := AC1 # !(D1) & x1 # l; "etape initiale x2 := AC2 & !l # !(D2) & x2 & !l; "etape 2 x3 := AC3 & !l # !(D3) & x3 & !l; "etape 3 act = x1 # x3; "equation combinatoire end </pre>	

2°) Vecteurs Tests

On ajoute au programme ci-dessus les vecteurs tests ci-dessous. Que donneront ces vecteurs tests ?

Données ↓	Réponses ↓
<pre> Test_vectors ([clock,e1,e2,e3,l]->[x1,x2,x3,act]) [c.,0,0,0,1]->[.X.,.X.,.X.,.X.]; [c.,0,0,0,0]->[.X.,.X.,.X.,.X.]; [c.,1,0,0,0]->[.X.,.X.,.X.,.X.]; [c.,0,1,0,1]->[.X.,.X.,.X.,.X.]; [c.,0,0,0,0]->[.X.,.X.,.X.,.X.]; [c.,1,0,0,0]->[.X.,.X.,.X.,.X.]; [c.,0,0,0,0]->[.X.,.X.,.X.,.X.]; [c.,0,1,0,0]->[.X.,.X.,.X.,.X.]; [c.,0,0,0,0]->[.X.,.X.,.X.,.X.]; [c.,0,0,1,0]->[.X.,.X.,.X.,.X.]; [c.,0,0,0,0]->[.X.,.X.,.X.,.X.]; </pre>	

3°) Donner les équations pour ce GRAFCET :

- de récurrence de l'étape 2
- de sortie de l'action act

Réponses pour les équations de récurrence :

- $x2+ =$
- $act =$

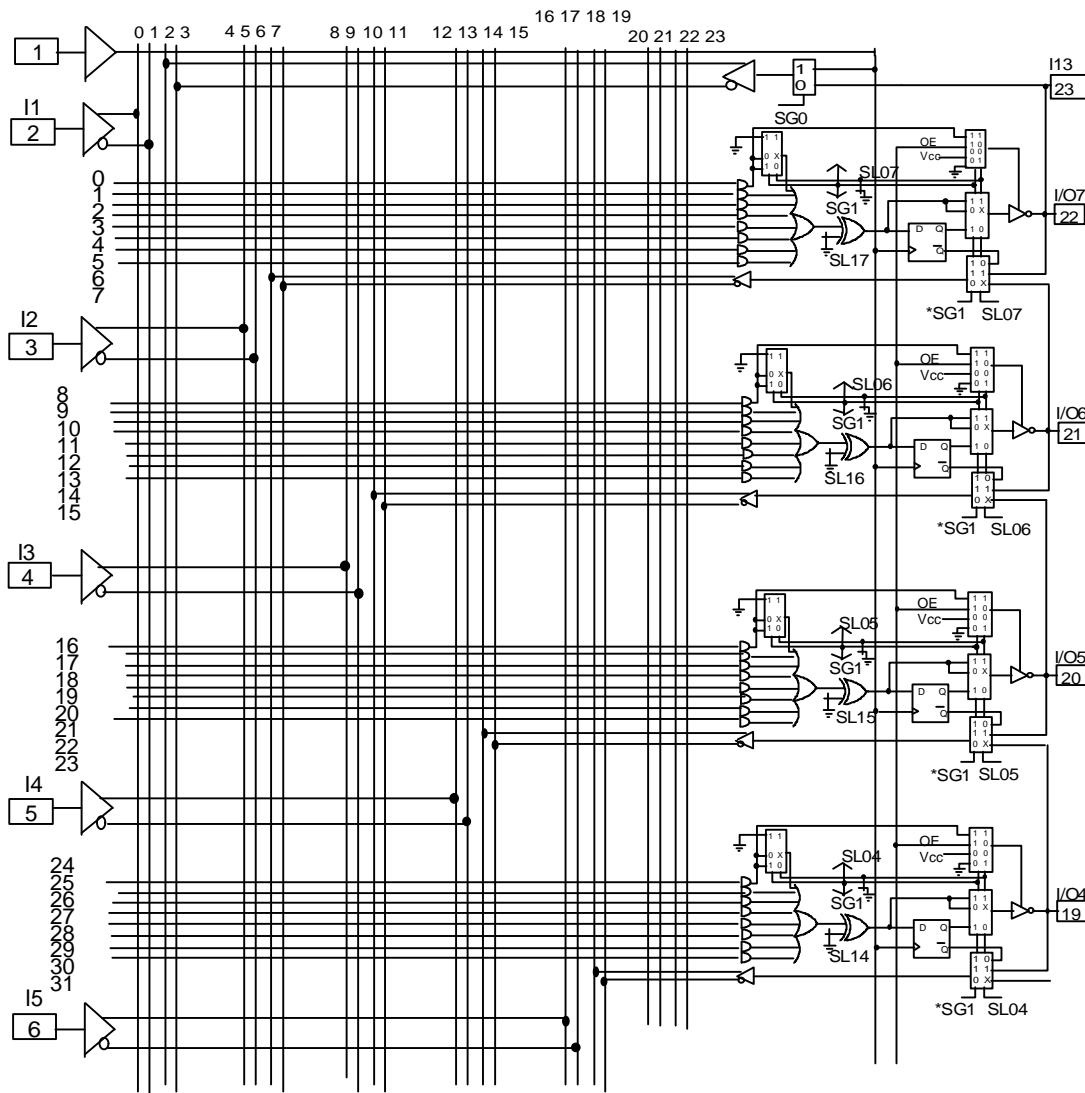
On vous demande d'implanter ces deux équations sur la GAL 20V8 de la page suivante en donnant les fusibles non grillés (croix) ainsi que la valeur des bits internes correspondants ci-dessous.

Réponses ici pour les bits internes :

SG ₁ =	SL0 ₅ =	SL1 ₅ =	<-- pour x2
SG1 =	SL0 ₇ =	SL1 ₇ =	<-- pour act

4°) Graphe d'état et programme ABEL correspondant. Pour le GRAFCET trouvé on vous demande de représenter dans le tableau ci-dessous son graphe d'état puis de compléter la partie du programme ABEL en state_diagram correspondant à ce graphe d'état pour les deux transitions manquantes (donc en laissant de côté les états inutilisés) et sans optimiser.

Graphe d'état : réponse : ↓	Partie du programme ABEL correspondant ↓
	<pre> state_diagram([x1,x2,x3] state [1,0,0] : if e1 then [0,1,0] else [1,0,0]; end </pre>



Ajouter sur la figure ci-dessus l'affectation (ou nom symbolique) des broches utilisées par ABEL pour les entrées / sorties.

TD 15 - Les opérations arithmétiques.

I - Addition binaire.

I-1) Principe :

Pour le système binaire, on définit l'addition comme pour le système décimal, en utilisant la table d'addition ci-contre.

	A	0	1	10
B	0	0	1	10
1	1	1	10	11
10	10	11	100	

Selon ce principe, l'addition de 2 bits a et b donne :
 - 1 bit S de somme (SUM en anglais)
 - 1 bit C de retenue (Carry).

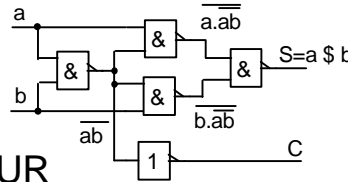
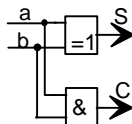
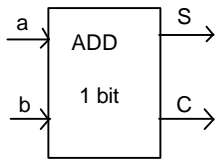
Recherchons leurs expressions

On peut donc réaliser cette addition sur un bit à l'aide des schémas logiques ci-dessous :

Le résultat (C,S) s'écrit donc en fonction de a et b :
 $S = a \oplus b$
 $C = a \cdot b$

	a	0	1
b	0	0	1
1	1	1	0
			S

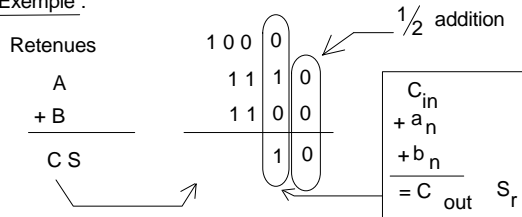
	b	0	1
a	0	0	0
1	1	0	1
			C



DEMI ADDITIONNEUR

C'est un **demi-additionneur**, en effet de façon plus générale l'addition s'effectue sur des nombres quelconques donc comportant plus de 2 bits.

Exemple :



- Carry in (Cin) est la retenue de l'addition des bits de rang précédent (r-1).
- Carry out (Cout) sera la retenue de l'addition des bits du rang (n).

L'addition s'effectue donc sur 3 bits et donne un bit de somme et un bit de retenue. Etablissons les expressions de S_n et C_{out} .

Table de vérité

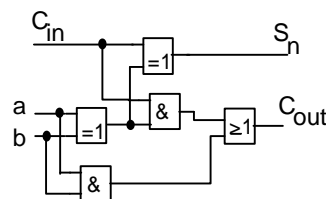
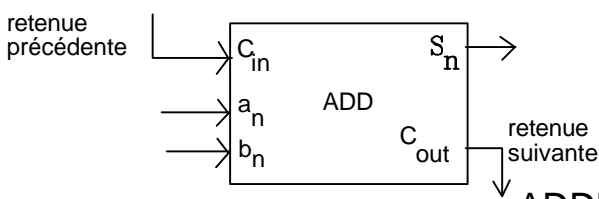
C_{in}	a_n	b_n	C_{out}	S_n
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

	ab	00	01	11	10
C_{in}	0	0	0	1	0
1	1	0	1	1	1
					C_{out}

	ab	00	01	11	10
C_{in}	0	0	1	0	1
1	1	0	1	1	0
					S_n

$$C_{out} = a \cdot b + C_{in} \cdot a + C_{in} \cdot b = a \cdot b + a \cdot b \cdot C_{in} + a \cdot b \cdot C_{in} \quad S_n = (a \oplus b) \oplus C_{in}$$

La réalisation de l'additionneur complet nécessite deux demi-additionneurs et peut se faire selon ces deux schémas :



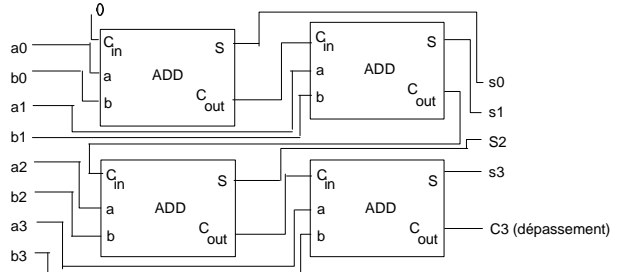
ADDITIONNEUR

Remarque : Dans les additionneurs intégrés, on cherche à minimiser le nombre de couches d'opérateurs logiques, c'est à dire le temps de réponse du circuit complet. On utilise pour cela d'autres écritures de S_n et C_{out} .

I- 2) Addition parallèle à retenue en cascade :

Additionneur à retenue propagée :

Ce circuit utilise n additionneurs complets. L'addition s'effectue sur les n bits simultanément mais en reportant la retenue de chaque calcul sur l'entrée C_{in} de l'entrée de l'additionneur suivant. Tout se passe comme si la retenue se propageait d'étage en étage.



ADDITIONNEUR A RETENUE PROPAGEE

Remarque :

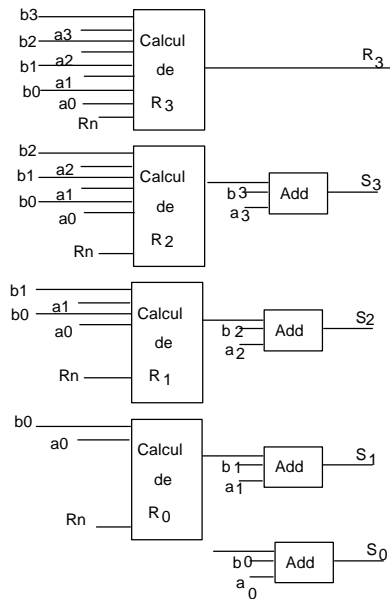
- 1) Si pour un additionneur le temps T_p de calcul de la retenue vaut $2t_p$ pour les n étages, le résultat ne s'affichera complètement (C_3 incluse) qu'au bout d'un temps $T=2.n.t_p$ (ici $8 t_p$), celui-ci pouvant être très long si n est grand.
- 2) Dans une représentation circulaire il y a dépassement $C_3=1$ si la somme $S>2n$.

Additionneur à retenue anticipée :

* Pour un additionneur complet de 1 bit nous avons vu que $C_{out}=ab+(a+b)C_{in}$ ceci signifie que :

- Si $a=b=1$ $C_{out}=1$ quel que soit C_{in} , il y a génération systématique d'une retenue C_{out} et l'on définit donc $G=a.b$ (**terme de génération**)
 - Si l'un des deux bits a ou b est à 1 et que $C_{in}=1$, il y a propagation de la retenue C_{in} et l'on définit $P=a+b$ (**terme de propagation**).
- Dans ces conditions, les équations de l'additionneur deviennent :

$C_{out} = G + P.C_{in}$ et $S=(a \text{ xor } b) \text{ xor } C_{in}$.



* le raisonnement s'applique pour un additionneur de deux nombres de 4 bits :

- Pour chaque rang, on crée les deux termes G_i et $P_i.r_{i-1}$ en posant $R=C_{out}$ et $r=C_{in}$

Pour chaque bit :

$$R_0 = G_0 + P_0.r_0$$

$$R_1 = G_1 + P_1.r_1 = G_1 + P_1.G_0 + P_1.P_0.r_0$$

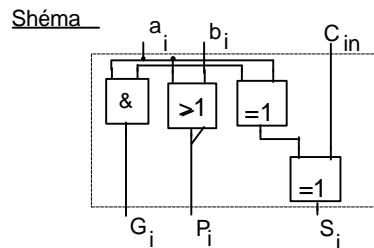
car $r_1 = R_0$

$$R_2 = G_2 + P_2.r_2$$

$$= G_2 + P_2.G_1 + P_2.P_1.G_0 + P_2.P_1.P_0.r_0$$

$$R_3 = G_3 + P_3.r_3 \text{ avec } r_3 = R_2$$

$$= G_3 + P_3.G_2 + P_3.P_2.G_1 + P_3.P_2.P_1.G_0 + P_3.P_2.P_1.P_0.r_0$$



- Pour chaque additionneur, on effectue le calcul de la retenue R_{i-1} et pour cela on leur associe un générateur de retenue anticipée (GRA).

Remarque : En observant la structure interne (page précédente), on constate que le calcul de R_i ne se fait qu'à travers 3 couches d'opérateurs seulement.
C'est sur ce principe qu'est réalisé le 7483, additionneur 4 bits.

II - La soustraction.

II-1) Principe (rappel)

* En binaire comme en arithmétique décimale, la soustraction $A-B$ s'effectue en ajoutant $(10)_2$ au chiffre de rang i (si nécessaire) puis en retranchant 1 de report (retenue) au chiffre b de rang suivant.

Exemple:

$$\begin{array}{r}
 \begin{array}{cccc}
 & (+10) & & (+10) \\
 A & & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\
 - B & & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\
 \hline
 = D & & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0
 \end{array}
 \end{array}$$

report

* Pour calculer la différence D , on utilise la méthode du complément vrai et on effectue donc une somme.

Exemple :

$ \begin{array}{r} A \quad 1011 \quad 1011 \\ - B \quad -0101 \Leftrightarrow +1011 \\ \hline = D \quad 0110 \quad \overset{1}{\leftarrow} \quad 0110 \\ \text{retenue} \quad \quad \quad S \\ \hline D = S = +0110 \end{array} $	$ \begin{array}{r} 0101 \quad 0101 \\ - 1011 \quad 0101 \\ \hline = D \quad 0101 \quad \overset{0}{\leftarrow} \quad 0101 \\ \text{retenue} \quad \quad \quad S \\ \hline D = -C_v S = -0110 \end{array} $
---	--

$A-B=D \Leftrightarrow A+C_v B=S$ avec $C_v B=C_r B+1$
Le résultat de cette dernière opération est toujours donné en complément vrai dans un format fixe.

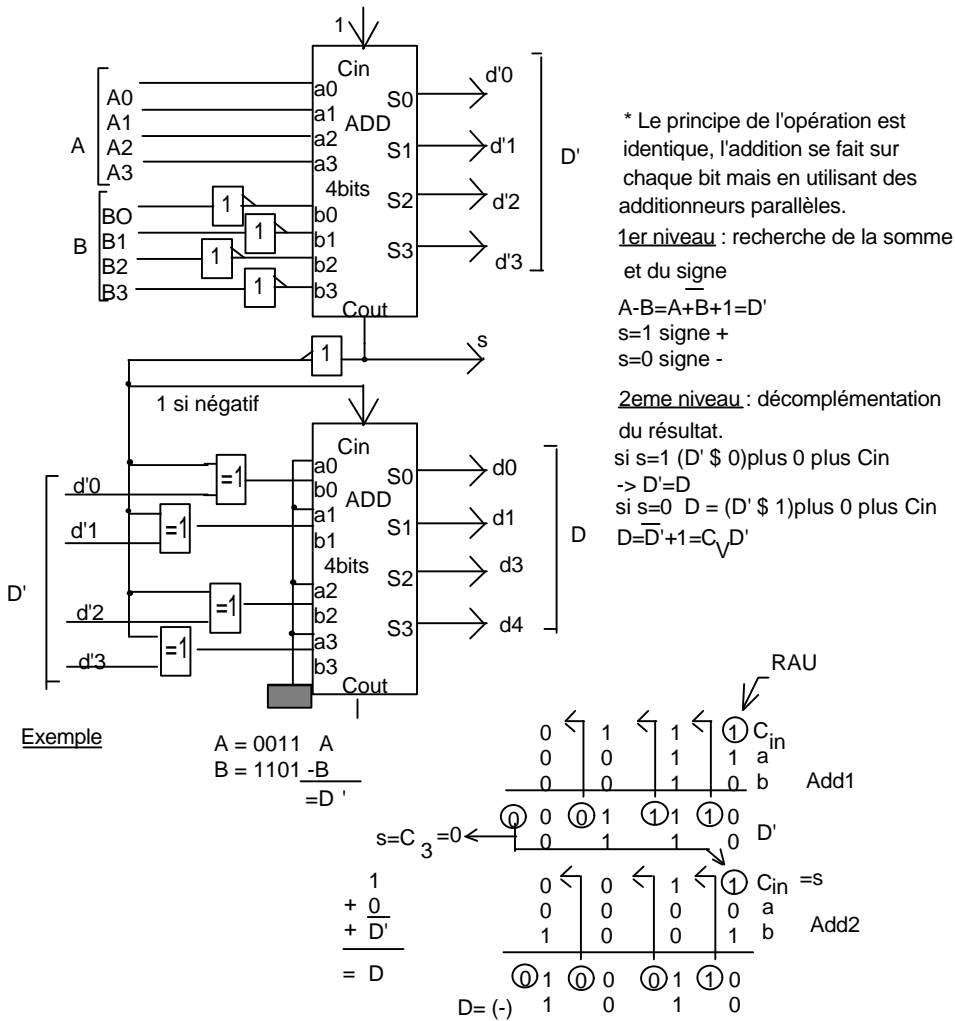
- si la **dernière retenue** de l'addition est 1 alors $D=S$; on néglige celle-ci et D est positif.
- si celle-ci **est égale à 0** alors $D=-C_v S$, on recherche le complément vrai en négligeant la dernière retenue.

Remarque : A n'est pas représenté en complément vrai.

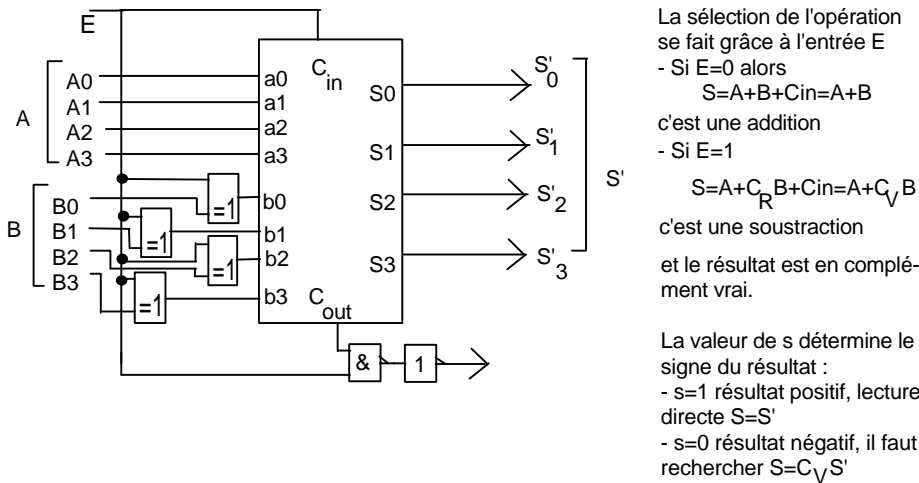
Remarque : le 4eme bit (à partir de la droite) donne le signes dans une représentation de nombre signé $s=0 S>0$ et $s=1 S<0$. Seuls les 3 premiers bits ont un sens.

II-2) Soustraction parallèle par complément vrai

Nous présentons ci-dessous à l'aide d'un schéma le principe du calcul.



II-3) Addition/soustraction en complément vrai.



III - Exercices - Additionneurs / Soustracteurs.

1) Construire un additionneur 1 bit à partir de deux demi-additionneurs.

2) Si le temps de propagation à travers une porte est t_p , calculer en fonction de t_p le temps de propagation d'un additionneur 4 bits à retenue propagée puis le temps de propagation d'un additionneur 4 bits à retenue anticipée.

3) On désire additionner deux chiffres codés BCD dont la somme est $\Sigma=A+B$. On dispose d'additionneurs binaires 4 bits effectuant la somme $(S)_2=(A)_2+(B)_2$.

a) Si on additionne A et B et que $S < 10$ on a alors $\Sigma=S$. Si $S \geq 10$, on constate que $\Sigma=S+n$.

Fixez-vous A et B, calculez S puis Σ et déterminez la valeur de n. Justifiez ce résultat.

b) Soit D le circuit de détection de $S \geq 10$. Recherchez l'équation de D. On dispose de 2 additionneurs binaires et de portes NAND. Proposez un schéma tel que $\Sigma=A+B$ pour $S < 10$ et que $\Sigma=A+B+n$ pour $S \geq 10$.

c) En prenant ensuite un exemple dont la somme est supérieure à 15 montrez qu'un circuit combinatoire doit être ajouté pour la retenue finale.

4) Etablir la table de vérité des fonctions D_i (différence) et c_{i+1} d'un soustracteur de nombres non signés à partir de a_i , b_i et c_i . Comparer D_i avec la fonction S_i d'un additionneur.

5) Les mots "addition parallèle" se retrouvent dans le cours. Il existe une autre version d'additionneur réalisant une addition série :

On donne ci-dessous deux schémas d'additionneur série. On vous demande combien de cycles d'horloge faut-il pour réaliser une opération complète sur n bits ?

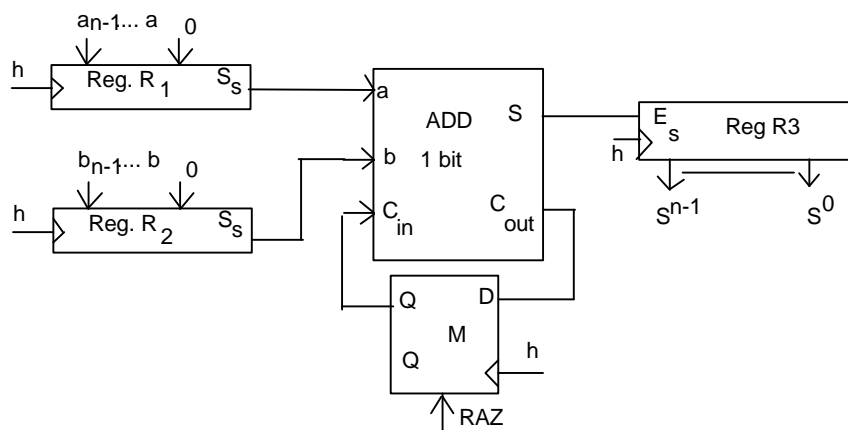
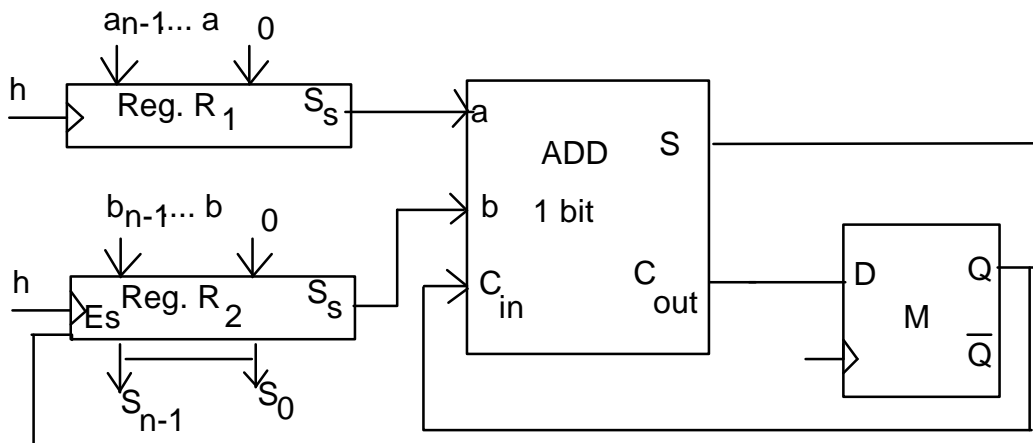


Schéma d'un additionneur série :

Additionneur série à deux registres :



Proposer le schéma d'un soustracteur série par complément vrai.

TD 16 - Comparaison, multiplication et calculette

I - Comparaison.

A partir d'un soustracteur, il est relativement aisé de concevoir un comparateur qui indiquera sur 3 fils de sorties soit $A < B$, soit $A = B$, soit $A > B$. Dans le premier cas en effet la retenue de gauche sera à 0, dans le dernier elle sera à 1 et dans le cas $A = B$ la soustraction de tous les bits de A et de B ne donnera que des 1 et 0 de retenue.

Il est naturellement possible d'implanter directement les équations logiques de comparaison, sans passer par la soustraction.

Le comparateur 7485 utilisé en TP (tachymètre) est un comparateur 4 bits binaire ou BCD.

4-BIT MAGNITUDE COMPARATOR

54/74 SERIES "85"

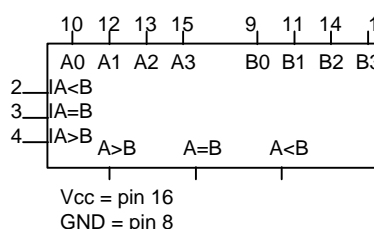
DESCRIPTION

The "85" is an expandable 4-bit Magnitude Comparator. It compares two 4-bit binary weighted words (A0-A3) and (B0-B3) where A3 and B3 are the most significant bits. The expansion inputs IA>B, IA=B, IA<B are treated as the least significant inputs for comparison purposes. The three active HIGH outputs are "A greater than B" (A>B) "A equals B" (A=B), and "A less than B" (A<B). These devices can be cascaded to almost any length.

FEATURES

- Magnitude comparison of any binary words
- Serial or parallel expansion without extra gating
- Use 54S/74S85 for very high speed comparisons

LOGIC SYMBOL



TRUTH TABLE

COMPARING INPUTS				CASCADING INPUTS			OUTPUTS		
A3,B3	A2,B2	A1,B1	A0,B0	IA>B	IA<B	IA=B	A>B	A<B	A=B
A3>B3	X	X	X	X	X	X	H	L	L
A3<B3	X	X	X	X	X	X	L	H	L
A3=B3	A2>B2	X	X	X	X	X	H	L	L
A3=B3	A2<B2	X	X	X	X	X	L	H	L
A3=B3	A2=B2	A1>B1	X	X	X	X	H	L	L
A3=B3	A2=B2	A1<B1	X	X	X	X	L	H	L
A3=B3	A2=B2	A1=B1	A0>B0	X	X	X	H	L	L
A3=B3	A2=B2	A1=B1	A0<B0	X	X	X	L	H	L
A3=B3	A2=B2	A1=B1	A0=B0	H	L	L	H	L	L
A3=B3	A2=B2	A1=B1	A0=B0	L	H	L	L	H	L
A3=B3	A2=B2	A1=B1	A0=B0	L	L	H	L	L	H
A3=B3	A2=B2	A1=B1	A0=B0	X	X	H	L	L	H
A3=B3	A2=B2	A1=B1	A0=B0	H	H	L	L	L	L
A3=B3	A2=B2	A1=B1	A0=B0	L	L	L	H	H	L

SIGNETICS

II - Multiplication

Pour imaginer un multiplieur binaire il faut analyser les lois d'obtention du résultat à partir des valeurs à multiplier. En décimal, la multiplication nécessite trois opérations :

- multiplication chiffre par chiffre nécessitant la connaissance des tables de multiplications,

- décalage des résultats successifs,
- addition des résultats partiels.

II-1) Table de multiplication binaire (à un bit).

Elle se résume à

0x0=0

0x1=0

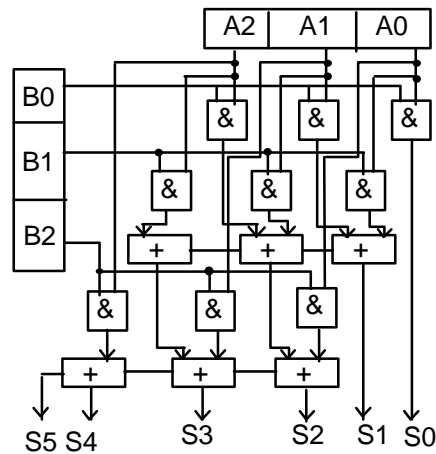
1x0=0

1x1=1

Elle est identique à la fonction ET.

II-2) Réalisation d'un multiplieur de deux nombres 3 bits

En implantant directement la "structure" même des opérations de multiplication, il est facile de concevoir un circuit permettant de multiplier A et B, chacun de ces nombres ayant 3 bits. Avec les décalages, il est facile d'imaginer que la résultat se fera sur 6 bits.



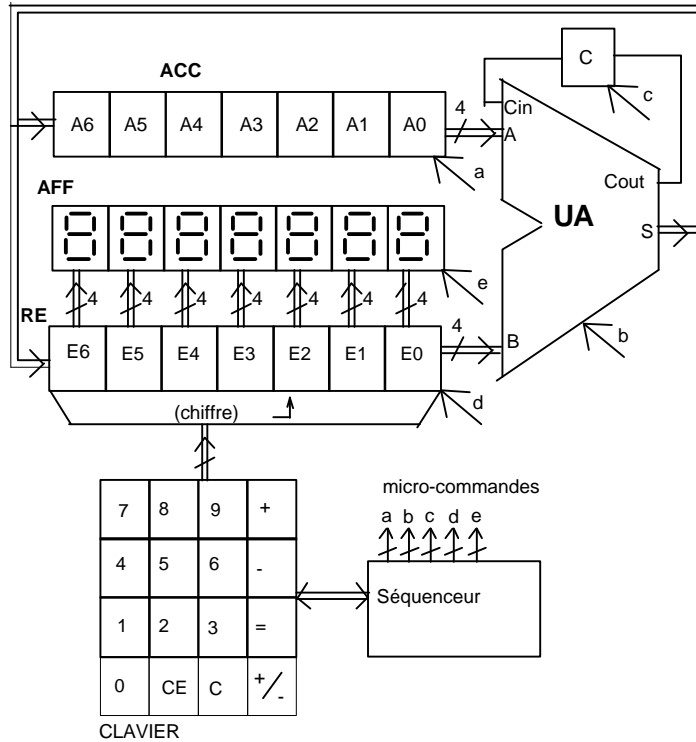
II-3) Circuit intégré.

Le circuit TTL 74274 est un multiplieur 4 bits basé sur ces principes. Il possède 8 bits de sortie protégés par des portes Trois états commandées par des broches /G1 et /G2.

III - Architectures d'une calculette simple

Il nous faut commencer par décrire la **ressource** c'est à dire l'ensemble des circuits électroniques commandés par le séquenceur.

- Unité arithmétique pouvant réaliser les opérations LDA, ADDC, SUBC RIEN. La commande "b" nécessite donc au moins 2 bits.
- C registre de retenue (Carry) ; commande "c" 1 bit.
- ACC registre à décalage 7x4 bits permettant un décalage à gauche, à droite une remise à zéro. Donc sa commande "a" comprend au moins 2 bits.
- RE identique à ci-dessus sauf commande "d".
- AFF 7 afficheurs ayant comme fonctionnalité : extinction, affichage, clignotement (dépassement) Sa commande "e" comprend donc au moins 2 bits.
- Le séquenceur permettant la scrutation du clavier et l'exécution des microcommandes "a", "b", "c", "d", "e".



IV - Exercices

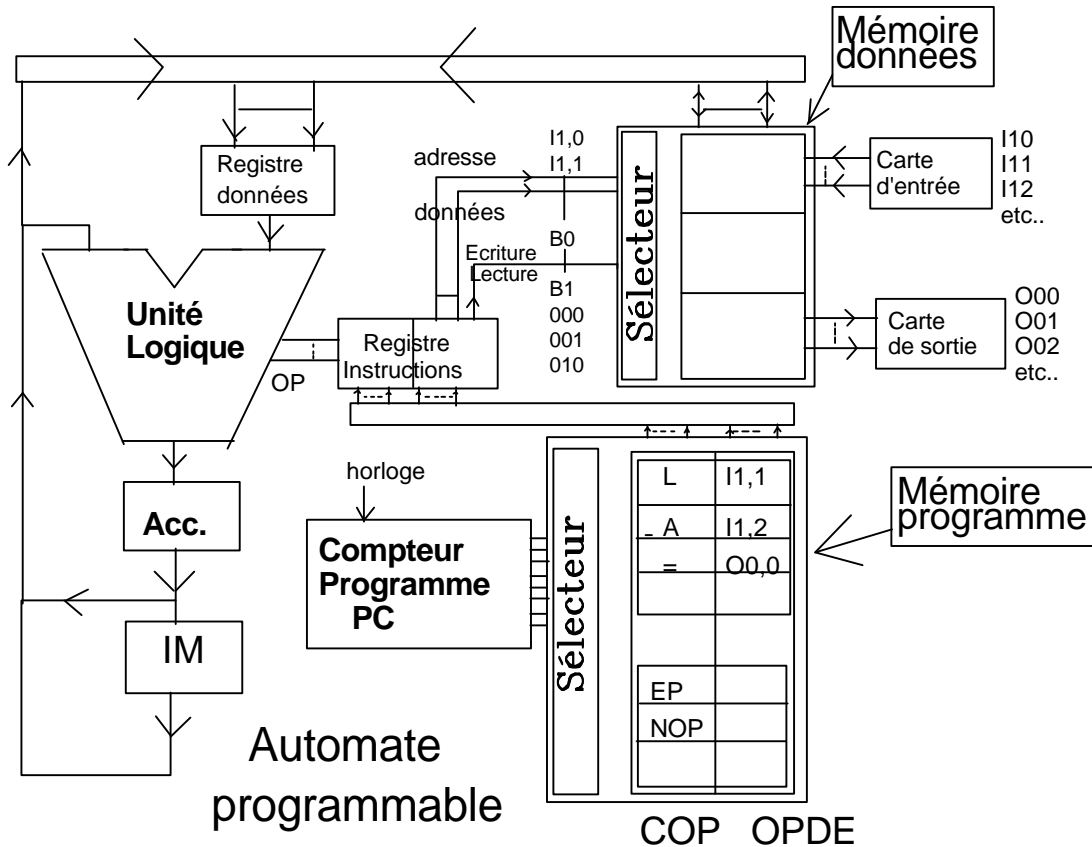
- 1) Exercice sur la soustraction BCD. Puis addition/soustraction pour l'UAL de la calculette.
- 2) Exercice sur la multiplication BCD. Peut-on faire simplement une modification de la calculette pour la faire multiplier ?

TD 17 - CEI 1131-3 : Le langage IL et LD

La norme CEI 1131-3 (Février 1993) normalise 5 langages :

- SFC (Sequentiel Function Chart) plus connu en France sous le nom de GRAFCET,
- LD (Ladder Diagramm) langage à contact,
- ST (Structured Text)
- IL (Instruction List)
- FBD (Function Block Diagram)

I - Architecture mémoire plus compteur plus UAL



II - Langage IL

II-1) La norme 1131-3 (Extraits choisis)

Il existe deux langages littéraux : ST et IL. Ces deux langages ont des éléments communs

II-1-a) Eléments communs (p226 de la norme)

Les éléments littéraux définis à l'article 2 doivent être communs aux langages littéraux (IL et ST) définis dans le présent article. En particulier, les éléments de structuration des programmes suivants doivent être communs aux langages littéraux :

```

TYPE...END_TYPE
VAR...END_VAR
VAR_INPUT...END_VAR
VAR_OUTPUT...END_VAR
VAR_IN_OUT...END_VAR
VAR_EXTERNAL...END_VAR
FUNCTION...END_FUNCTION
    
```

FUNCTION_BLOCK...END_FUNCTION_BLOCK
 PROGRAM...END_PROGRAM
 STEP...END_STEP
 INITIAL_STEP...END_STEP
 TRANSITION...END_TRANSITION
 ACTION...END_ACTION

II-1-b) Langage IL (Liste d'instructions) (p226 de la norme)

Le présent paragraphe définit la sémantique du langage IL (Liste d'instructions) dont la syntaxe formelle est donnée un peu plus loin.

- Instructions

Comme l'indique le Tableau 1, une liste d'instructions est composée d'une suite d'instructions. Chaque instruction doit débiter sur une nouvelle ligne et doit contenir un opérateur accompagné de modificateurs optionnels et, si cela est nécessaire pour l'opération considérée, un ou plusieurs opérandes séparés par des virgules. Les opérandes peuvent être choisis parmi les représentations de données définies en 2.2 pour les libellés et en 2.4 pour les variables.

L'instruction peut être précédée d'une étiquette d'identification suivie de deux points (:). Si un commentaire tel que défini en 2.1.5 est présent, il doit constituer le dernier élément d'une ligne. Des lignes vides peuvent être insérées entre les instructions.

Tableau 1 - Exemples de champs d'instructions

Etiquette	Opérateur	Opérande	Commentaire
START:	LD	%IX1	(* BOUTON POUSSOIR *)
	ANDN	%MX5	(* NON INHIBEE *)
	ST	%QX2	(* MARCHE VENTILATEUR *)

- Opérateurs, modificateurs et opérandes (p228 de la norme)

Les opérateurs standards ainsi que leurs modificateurs et opérandes autorisés doivent être tels qu'énumérés dans le tableau 2.

.....

Le modificateur "N" indique une négation booléenne de l'opérande.

Par exemple l'instruction ANDN %IX2 est interprétée de la manière suivante :

resultat := resultat AND NOT %IX2

Le modificateur parenthèse gauche "(" indique que l'évaluation de l'opérateur est différée jusqu'à ce qu'un opérateur parenthèse droite ")" soit rencontré ; par exemple la séquence d'instructions suivante:

AND(%IX1
 OR %IX2
)

doit être interprétée de la manière suivante :

resultat := resultat AND (%IX1 OR %IX2)

Le modificateur "C" indique que l'instruction donnée ne doit être exécutée que si le résultat faisant l'objet de l'évaluation en cours a la valeur booléenne 1 (ou la valeur booléenne 0 si l'opérateur est combiné avec le modificateur "N").

Tableau 2 - Opérateurs de liste d'instructions (p 230 de la norme)

N°	Opérateur	Modificateur	Opérande	Sémantique
1	LD	N	Note 2	Rendre le résultat courant égal à l'opérande
2	ST	N	Note 2	Mémoriser le résultat à l'emplacement de l'opérande
3	S	Note 3	BOOL	Positionner l'opérande booléen à 1
	R	Note 3	BOOL	Remettre l'opérande booléen à 0

4	AND	N, (BOOL	AND booléen
5	&	N, (BOOL	AND booléen
6	OR	N, (BOOL	OR booléen
7	XOR	N, (BOOL	OR exclusif booléen
8	ADD	(Note 2	Addition
9	SUB	(Note 2	Soustraction
10	MUL	(Note 2	Multiplication
11	DIV	(Note 2	Division
12	GT	(Note 2	Comparaison : >
13	GE	(Note 2	Comparaison : >=
14	EQ	(Note 2	Comparaison : =
15	NE	(Note 2	Comparaison : <>
16	LE	(Note 2	Comparaison : <=
17	LT	(Note 2	Comparaison : <
18	JMP	C, N	LABEL	Saut vers l'étiquette
19	CAL	C, N	NAME	Appel d'un bloc fonctionnel (Note 4)
20	RET	C, N		Retour d'une fonction appelée ou d'un bloc fonctionnel
21)			Evaluation d'une opération différée

NOTES
 1 Se reporter au texte précédent pour toute explication relative aux modificateurs et à l'évaluation des expressions.
 2 Ces opérateurs doivent être soit surchargés soit saisis comme défini en 2.5.1.4. Le résultat courant et l'opérande doivent être du même type
 3 Ces opérations sont effectuées si et seulement si le résultat courant a la valeur booléenne 1
 4 Le nom du bloc fonctionnel est suivi par une liste d'arguments entre parenthèses, tels que définis en 3.2.3.
 5 Lorsqu'une instruction JMP est contenue dans une construction ACTION...END_ACTION l'opérande doit être une étiquette à l'intérieur de la même construction

- Fonctions et blocs fonctionnels (p 230 de la norme)

Les fonctions doivent être lancées en inscrivant le nom de la fonction dans le champ opérateur. Le résultat courant doit être utilisé comme premier argument de la fonction. Si nécessaire les arguments supplémentaires doivent être fournis dans le champ opérande. La valeur renvoyée par une fonction, après exécution correcte d'une instruction RET ou lorsque la fin physique d'une fonction est atteinte doit devenir le "résultat courant".

Les blocs fonctionnels peuvent être lancés sous condition ou inconditionnellement à l'aide de l'opérateur CAL (Appel) figurant dans le tableau 2. Comme l'illustre le tableau 3, ce lancement peut prendre trois formes. Les opérateurs d'entrée indiqués dans le tableau 4 peuvent être utilisés conjointement à la caractéristique 3 du tableau 3.

Tableau 3 - Caractéristiques du lancement de bloc fonctionnel en langage IL (p 232)

N°	Description / exemple
1	CAL avec liste d'entrées CAL C10 (CU := %IX10, PV:=15)
2	CAL avec entrées de charge/mémoire LD 15 ST C10.PV LD %IX10 ST C10.CU CAL C10
3	Utilisation d'opérateurs d'entrée: LD 15 PV C10 LD %IX10 CU C10

NOTE - Une déclaration telle que VAR C10 : CTU; END_VAR est supposée dans les

exemples ci-dessus

- Durée (p 52 de la norme)

Les données relatives à la durée doivent être délimitées à gauche par le mot clef T#, TIME#, t# ou time#. La représentation des données relatives à la durée, en termes de jours, heures, minutes, secondes et millisecondes, ou toute combinaison de ces derniers, doit être pris en charge conformément au tableau 5. L'unité de temps la moins significative peut être écrite en notation réelle sans exposant.

Les unités des libellés peuvent être séparés par des caractères de soulignement.

Le "dépassement" de l'unité la plus significative d'un libellé de temps est permis; par exemple la notation T#25h_15m est permise.

Les unités de temps, comme par exemple secondes, millisecondes, etc. peuvent être représentées en majuscules et en minuscules.

Tableau 5 - Caractéristique de libellés de temps

N°	Description de la caractéristique	Exemples
1a	Libellés de temps sans caractères de soulignement :	T#14ms T#14.7s T#14.7m
	Préfixe court	T#14.7h t#14.7d t#25h15m
1b	Préfixe long	t#5d14h12m18s3.5ms
2a	Libellés de temps avec caractère de soulignement :	TIME#14ms time#14.7s
	Préfixe court	t#25h_15m t#5d_14h_12m_18s_3.5ms
2b	Préfixe long	TIME#25h_15m time#5d_14h_12m_18s_3.5ms

- Types de données élémentaires (p 56 de la norme)

Les types de données élémentaires, le mot clef relatif à chaque type de donnée, le nombre de bits par élément d'information, et la plage de valeurs relative à chaque type de données élémentaire doivent être tels que définis dans le tableau 6.

Tableau 6 - Types de données élémentaires

N°	Mot clef	Type de donnée	Bits	Etendue
1	BOOL	Booléen	1	Note 8
2	SINT	Entier court	8	Note 2
3	INT	Entier	16	Note 2
4	DINT	Entier double	32	Note 2
5	LINT	Entier long	64	Note 2
6	USINT	Entier cour non signé	8	Note 3
7	UINT	Entier non signé	16	Note 3
8	UDINT	Entier double non signé	32	Note 3
9	ULINT	Entier long non signé	64	Note 3
10	REAL	Nombre réel	32	Note 4
11	LREAL	Réels longs	64	Note 5
12	TIME	Durée	Note 1	Note 6
13	DATE	Date (uniquement)	Note 1	Note 6
14	TIME_OF_DAY ou TOD	Heure du jour (uniquement)	Note 1	Note 6
15	DATE_AND_TIME ou DT	Date et heure du jour	Note 1	Note 6
16	STRING	Cordon de caractères de longueur variable	Note 1	Note 7
17	BYTE	Cordon de bits de longueur 8	8	Note 7
18	WORD	Cordon de caractères de longueur 16	16	Note 7
19	DWORD	Cordon de caractères de longueur 32	32	Note 7
20	LWORD	Cordon de caractères de longueur 64	64	Note 7
NOTES				
1 La longueur de ces données dépend de l'application concernée.				

- 2 L'étendue des valeurs, relatives à des variables de ce type de donnée, est comprise entre - $(2^{**}(\text{Bits}-1))$ et $(2^{**}(\text{Bits}-1))-1$.
- 3 L'étendue des valeurs, relatives à des variables de ce type de donnée, est comprise entre 0 et $2^{**}(\text{Bits})-1$.
- 4 L'étendue des valeurs, relatives à des variables de ce type de donnée, doit être telle que définie dans la norme CEI 559 pour le format virgule flottante à largeur binaire.
- 5 L'étendue des valeurs, relatives à des variables de ce type de donnée, doit être telle que définie dans la norme CEI 559 pour le format virgule flottante à double largeur binaire.
- 6 L'étendue des valeurs, relatives à des variables de ce type de donnée, dépend de l'application concernée.
- 7 Une étendue numérique de valeurs ne s'applique pas à ce type de donnée.
- 8 Les valeurs que peut prendre variables de ce type de donnée doivent être 0 et 1, correspondant respectivement aux mots clés FAUX et VRAI.

Remarques : on peut se demander qui a traduit aussi mal cette norme !!!!!

- Pour les n° 18, 19 et 20, il s'agit de cordon de bits (Bit string) et non pas de cordon de caractères
- Le mot cordon n'est d'ailleurs pas approprié on parle plutôt habituellement de chaîne.
- Dans la note 8 sont traduits des mots clés ce qui est impossible !!!!

- Les opérandes sont :

(Tableau 7 p70 de la norme)

N°	Préfixe	Signification
1	I	Emplacement d'entrée
2	Q	Emplacement de sortie
3	M	Emplacement de mémoire
4	X	Taille d'un seul bit
5	Aucun	Taille d'un seul bit
6	B	Taille d'un octet (8bits)
7	W	Taille d'un mot (16 bits)
8	D	Taille d'un double mot (32 bits)
9	L	Taille d'un mot long (Quad) (64 bits)

NOTES

1 Sauf déclaration contraire, le type de donnée d'une variable directement adressable, de la taille d'un "seul bit" doit être BOOL.

2 Les organismes nationaux de normalisation peuvent publier des tables de traduction de ces préfixes.

- On peut résumer le langage IL par sa description formelle : sa grammaire (p 292 de la norme).

REGLES DE PRODUCTION:

```

instruction_list ::= instruction {instruction}
instruction ::= [[label:'] (il_operation | il_fb_call)] EOL
label ::= identifieur
il_operation ::= il_operator [' ' il_operand_list]
il_operand_list ::= il_operand {' ' il_operand}
il_operand ::= [identifieur ':=' ] (constant | variable)
il_fb_cal ::= 'CAL' ['C'['N']] fb_name ['(' il_operand_list ')']
il_operator ::= ('LD' | 'ST ') ['N'] | 'S' | 'R'
                | ('AND' | 'OR' | 'XOR') ['N'] ['(']
                | ('ADD' | 'SUB' | 'MUL' | 'DIV') ['(']
                | ('GT' | 'GE' | 'EQ' | 'LT' | 'LE') ['(']
                | ('JMP' | 'RET') ['C' ['N']]
                | 'S1' | 'R1' | 'CLK' | 'CU' | 'CD' | 'PV'
                | 'IN' | 'PT' | ')'
                | function_name
identifieur ::= (lettre - ('_'(lettre | chiffre))) {'_'}(lettre | chiffre)
    
```

II -2) Exemple du TSX 37 (Telemecanique)

- instructions de tests : LD, LDN, LDR, LDF, AND, ANDN, ANDR, ANDF, OR, ORN, ORR, ORF, XOR, XORN, XORR, XORF, AND(, AND(N, AND(R, AND(F, OR(, OR(N, OR(R, OR(F. On peut donc utiliser des parenthèses avec éventuellement des modificateurs : N, F (Falling edge), R (Rising edge) des blocs entre crochets [et].

Remarque : Dans la norme 1131-3 n'apparaissent pas les modificateurs R et F mais des blocs fonctionnels de détection de front montants et de fronts descendants (p140 de la norme). On pourrait expliquer LDR, LDF, ANDR, ANDF ... comme fonctions prédéfinies par Telemecanique. Mais il ne peut en être de même pour AND(R, ... Il semblerait que Telemécanique ait pris quelques libertés ici.

- instructions d'actions : ST, STN, R, S
- instructions de contrôles : JMP, JMPC, JMPCN, SRn, RET, RETC, RETCN, END, ENDC, ENDCN, HALT (END ... HALT n'apparaissent pas dans la norme 1131-3 et SRn s'appelle CAL)

On dispose de trois instructions de manipulation de pile:

- MPS (Memory PuSh) stocke l'accumulateur sur la pile
- MRD (Memory ReaD) lit le sommet de la pile
- MPP (Memory PoP) lit en déstockant le sommet de la pile

Ces instructions sont hors norme 1131-3 et on ne peut pas considérer qu'il s'agit de fonctions car ils ne satisfont pas la règle de la norme (p 84) :

"Les fonctions ne doivent comporter aucune information concernant les états internes, c'est à dire que le lancement d'une fonction dotée des mêmes arguments (paramètres d'entrée) doit toujours donner la même valeur"

Quand aux blocs fonctionnels ils doivent satisfaire la règle (norme p 118)

" Toutes les valeurs des variables de sortie et des variables internes nécessaires de cette structure de données doivent persister, d'un bloc fonctionnel au suivant; par conséquent, il n'est pas toujours nécessaire que le lancement du même bloc fonctionnel ayant les mêmes arguments (paramètres d'entrée) aboutissent aux mêmes valeurs de sortie"

Le problème c'est que selon la norme l'appel se fait alors avec un CAL (voir II-1-b ci-dessus)

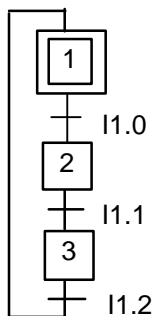
Les opérandes sont :

- des entrées notées %I1.0, %I1.1, (avec le signe %)
- des sorties notées %Q2.0, %Q2.1,..... (avec le signe %)
- des bits internes notés %M0, %M1.... (avec le signe %)
- des bits systèmes notés %Si (avec le signe %) S0 est le bit système de reprise à froid et S1 le bit système de reprise à chaud.
- des bits extraits de mots notés MWi:Xk (kième bit du ième mot)
- des bits de bloc de fonction.

II - 3) Programmation de GRAFCETs en langage IL

Il existe deux méthodes pour résoudre ce problème. A noter que des ateliers comme ISaGRAF qui se disent conformes à la norme ne proposent pas la deuxième méthode ci-dessous.

Première méthode : on utilise les équations de récurrence.



```

LD %S1 (*M0 == étape 1 *)
OR(%M2 (*M2 == étape 3 *)
AND %I1.2
)
S %M0
R %M2
LD %M1 (*M1 == étape 2*)
AND %I1.1
S %M2
R %M1
....
  
```

Deuxième méthode : des éléments prédéfinis existent :

- construction STEP ... END_STEP ou INITIAL_STEP ... END_STEP
- construction TRANSITION ... END_TRANSITION (TRANSITION FROM et TO)

- construction ACTION ...END_ACTION

Exemples : (tirés de la norme p 160)

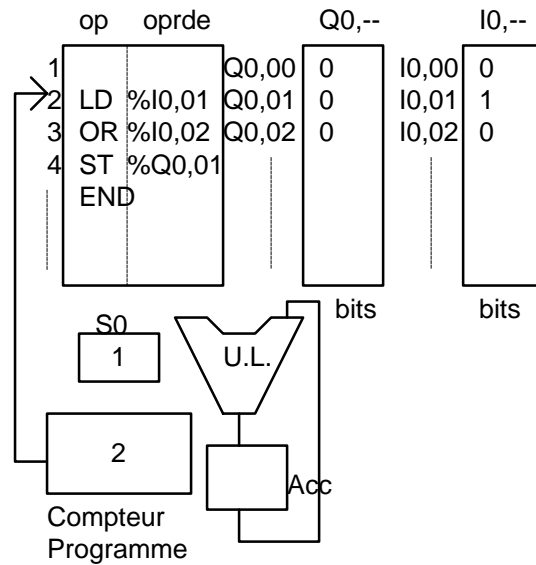
Langage IL	Langage ST
<pre>STEP STEP7 : A(N);END_STEP (*action A normale *) TRANSITION FROM STEP7 TO STEP 8 : LD %IX2.4 AND %IX2.3 END_TRANSITION STEP STEP8 : B(S);END_STEP (* action SET B*) STEP STEP7 : A(N);B(L,t#10s); END_STEP TRANSITION FROM (STEP7,STEP8) TO STEP 9 : LD %IX2.4 AND %IX2.3 END_TRANSITION STEP STEP9 : B(S);A(N);END_STEP (*deux actions*)</pre>	<pre>STEP STEP7 : END_STEP TRANSITION FROM STEP7 TO STEP 8 : := %IX2.4 & %IX2.3; END_TRANSITION STEP STEP8 : END_STEP</pre>

IV - Exercices

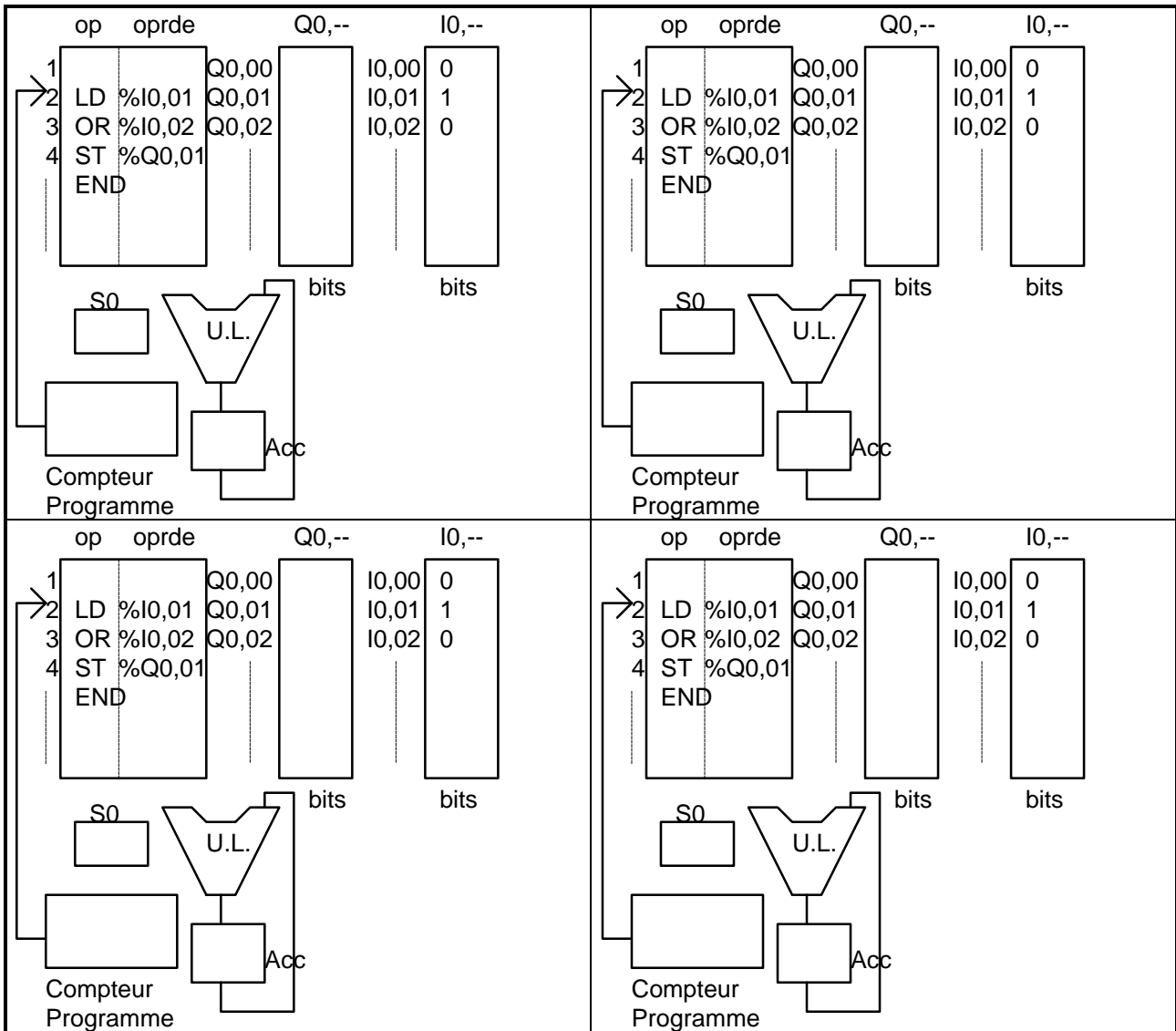
Exercice 1

L'état interne d'un automate de style TSX37 est donné ci-contre. Donnez sur les schémas suivants l'ensemble des modifications lors du déroulement du programme complet.

S0 bit système de reprise à froid
S1 bit système de reprise à chaud



Réponse :



Exercice 2

Le langage IL admet l'utilisation de parenthèses. Modifier l'architecture de l'exercice précédent pour pouvoir exécuter un programme avec parenthèses. Ne pas oublier que l'on peut faire plusieurs niveaux de parenthèses.

Exercice 3

La norme 1131-3 prévoit certains blocs fonctionnels prédéfinis :

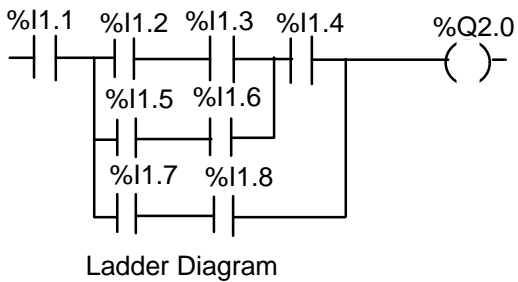
R_TRIG	entrée CLK et sortie Q	détection de front montant
F_TRIG	entrée CLK et sortie Q	détection de front descendant
CTU	entrées CU,R booléens entrée PV INT (valeur maximale) sortie Q booléenne sortie CV INT (valeur en cours)	compteur (ne fonctionnant pas sur front!)

1°) Réaliser en IL un compteur de 0 à 10 sur front.

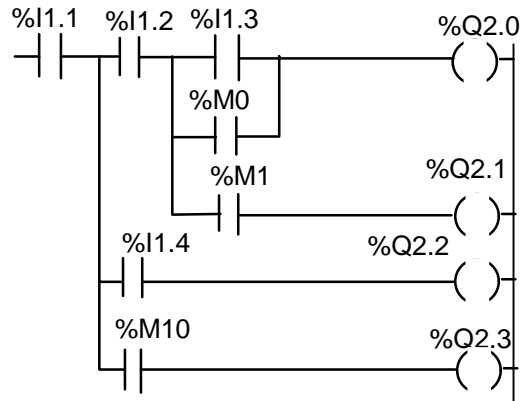
2°) Refaire le programme en utilisant les spécificités telemecanique.

Exercice 4

Programmer le réseau ci-dessous avec le langage IL et l'imbrication de parenthèses



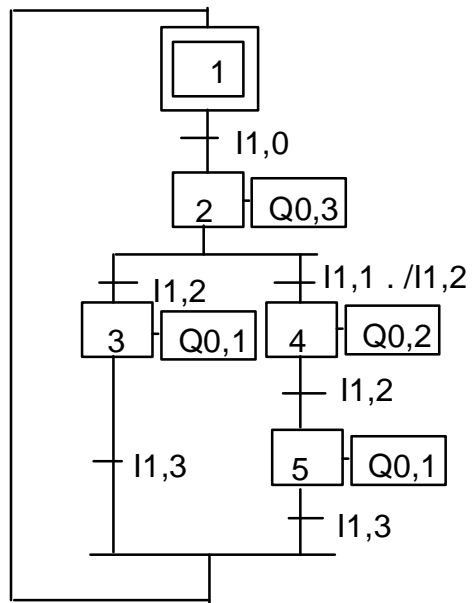
Programmer le réseau ci-dessous avec le langage IL et les instructions de pile



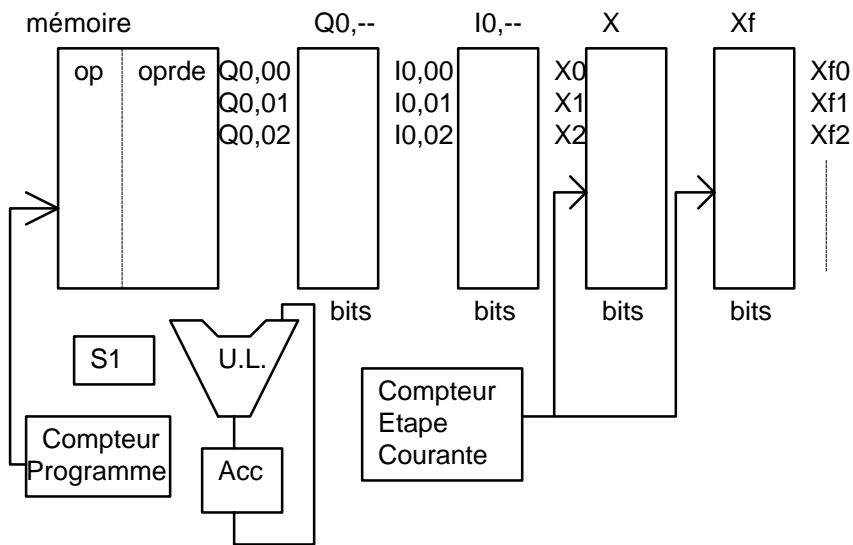
Exercice 5

Questions

- 1°) Reprendre le GRAFCET ci-contre et le programmer en langage IL en utilisant un bit mémoire par étape.
- 2°) Programmer ce GRAFCET avec les STEP... END_STEP etc..
- 3°) Montrer que ce dernier genre de programme nécessite une architecture comme ci-dessous



Architecture interne du TSX 37



TD 18 - CEI 1131-3 Langage SFC et ST

SFC (Sequentiel Function Chart) est un langage de programmation graphique qui ressemble au GRAFCET. La programmation des actions et transitions doit être examinée.

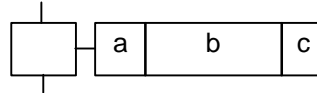
I) Programmation des actions

Norme CEI 848 (1988) complétée par 1131-3 (1993)

I-1) Généralités sur les actions

Ordre (action) détaillée :

- La section a contient une lettre symbole ou une combinaison de lettres symboles décrivant comment le signal binaire de l'étape sera traité.
- La section b contient une déclaration symbolique ou littérale décrivant l'ordre ou l'action.
- La section c indique le numéro de référence du signal de fin d'exécution correspondant



NOTES

1. La section b doit être au moins deux fois plus grande que les sections a et c.
2. Les sections a et c ne sont spécifiées que si nécessaire.

I-2) Les différentes actions

Le niveau 2 d'une étape SFC est la description détaillée des actions exécutées quand l'étape est active. Cette description est réalisée à l'aide des structures littérales du SFC, et des autres langages, tels que le ST (Structured Text). Voici les différents types d'action :

I-2-a) Actions booléennes

Une action booléenne consiste à assigner à une variable booléenne le signal d'activité d'une étape (drapeau d'étape). La variable booléenne doit être interne ou de sortie. Elle est forcée à chaque fois que le signal d'activité de l'étape change d'état. Voici la syntaxe des actions booléennes non mémorisées:

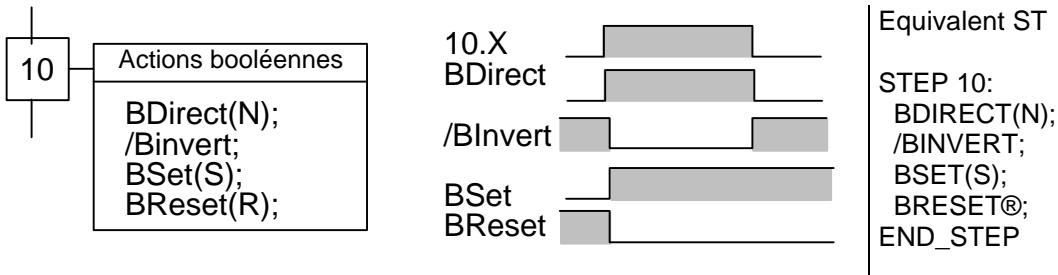
```
<variable_bool> (N);   copie le signal d'activité de l'étape dans la variable
<variable_bool> ;     même effet (l'attribut N est optionnel)
/ <variable_bool> ;   copie l'inverse logique du signal d'activité de l'étape dans la variable
```

D'autres types d'actions consistent à forcer la variable (à faux ou à vrai) quand l'étape devient active. Voici la syntaxe de ces actions :

```
<variable_bool> (S) ;  force la variable à TRUE quand le signal d'activité de l'étape prend l'état TRUE
<variable_bool> (R) ;  force la variable à FALSE quand le signal d'activité de l'étape prend l'état TRUE
```

On peut trouver les qualificatifs d'action suivants : Néant, N, R, S, L, D, P, SD, DS et SL (Norme p172).

La variable booléenne doit être INTERNE ou de SORTIE. La programmation SFC suivante représente le comportement indiqué dans le chronogramme:



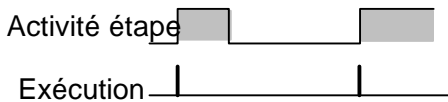
I-2-b) Actions impulsives

Une action impulsive est une liste d'instructions ST ou IL, exécutée une seule fois quand l'étape devient active. Les instructions sont écrites selon la syntaxe SFC suivante :

```

ACTION (P) :
    (* Enoncés ST *)
END_ACTION ;
    
```

Voici le chronogramme d'une action de type P :



I-2-c) Actions non mémorisées

Une action normale (ou non mémorisée) est une liste d'instructions ST ou IL, exécutée à chaque cycle pendant toute la durée d'activité de l'étape. Les instructions sont écrites selon la syntaxe SFC suivante :

```

ACTION (N) :
    (* Enoncés ST *)
END_ACTION ;
    
```

I-2-d) Actions SFC

Une action SFC est une séquence fille SFC, lancée ou tuée selon les évolutions du signal d'activité de l'étape. Une action SFC peut être décrite avec les qualificatifs d'action N (Non mémorisée), S (Set), ou R (Reset). Voici la syntaxe des actions SFC :

```

<progr_fils> (N) ;      lance la séquence fille quand l'étape devient active, et la tue lorsque l'étape
redevient inactive
<progr_fils> ;        même effet (l'attribut N est optionnel)
<progr_fils> (S) ;    lance la séquence fille quand l'étape devient active.
rien n'est fait lorsque l'étape redevient inactive
<progr_fils> (R) ;    tue la séquence fille quand l'étape devient active.
rien n'est fait lorsque l'étape redevient inactive
    
```

La séquence SFC spécifiée dans l'action doit être un programme SFC fils du programme en cours d'édition.

I-2-e) Appel de fonction ou de bloc fonctionnel

Les sous-programmes, fonctions ou blocs fonctionnels (écrits en ST, IL, LD ou FBD), ou les fonctions ou blocs fonctionnels "C" peuvent être directement appelés depuis un bloc d'action, selon la syntaxe suivante :

Pour les sous-programmes, fonctions ou fonctions "C":

```

ACTION (P) :
    
```

```

    <resultat> := <sous_programme> ( ) ;
END_ACTION;

```

ou

```

ACTION (N) :
    <resultat> := <sous_programme> ( ) ;
END_ACTION;

```

Pour les blocs fonctionnels écrits en "C" ou en ST, IL, LD, FBD:

```

ACTION (P) :
    Fbinst(in1, in2);
    result1 := Fbinst.out1;
    result2 := Fbinst.out2;
END_ACTION;

```

ou

```

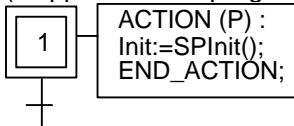
ACTION (N) :
    Fbinst(in1, in2);
    result1 := Fbinst.out1;
    result2 := Fbinst.out2;
END_ACTION;

```

Vous trouverez la syntaxe détaillées de ces appels dans le chapitre concernant le langage ST.

Exemple d'appel de sous-programme dans un bloc d'action :

(* Appel de sous-programme dans une action SFC*)

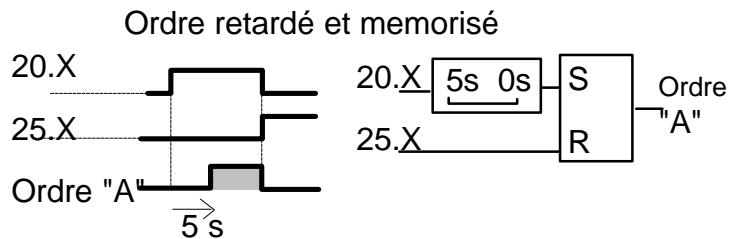
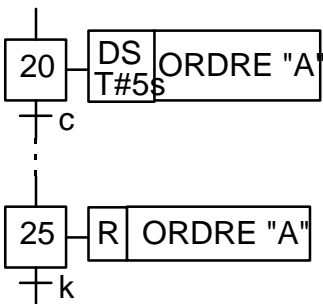
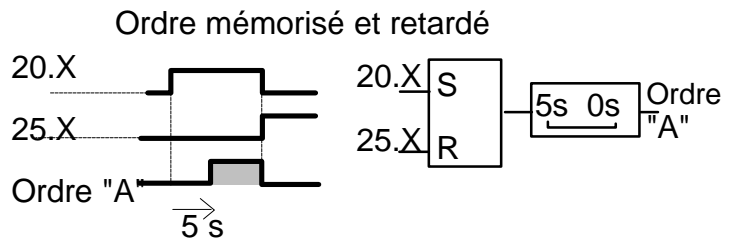
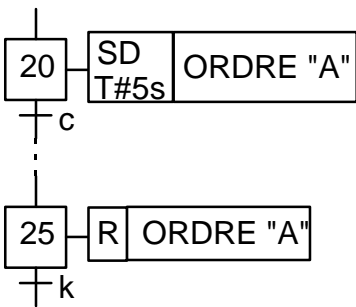
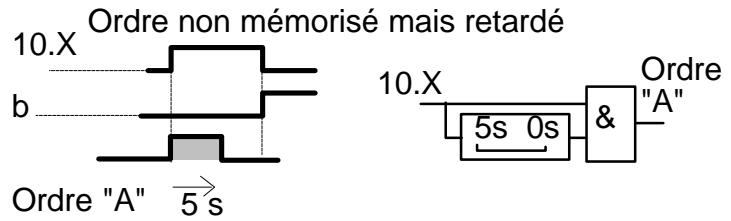
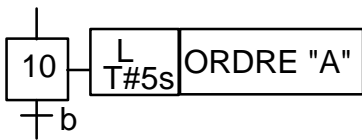
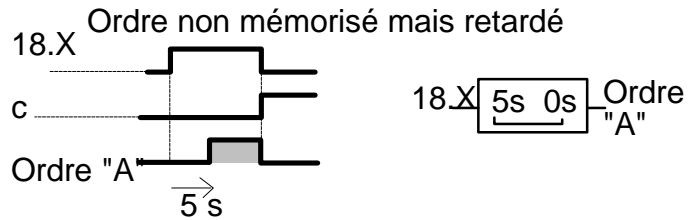
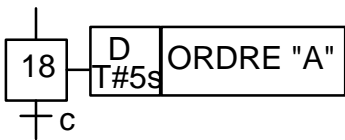
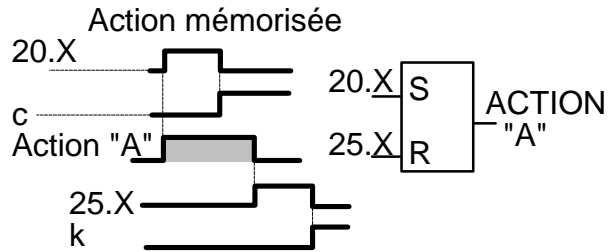
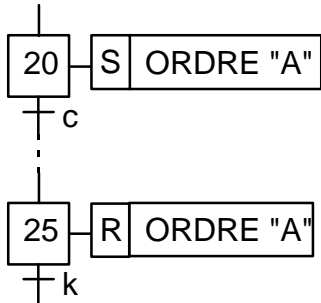
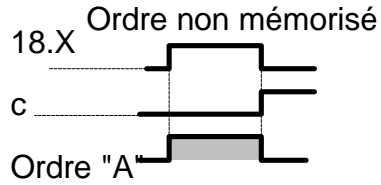
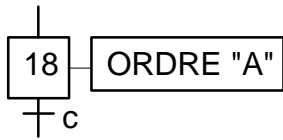


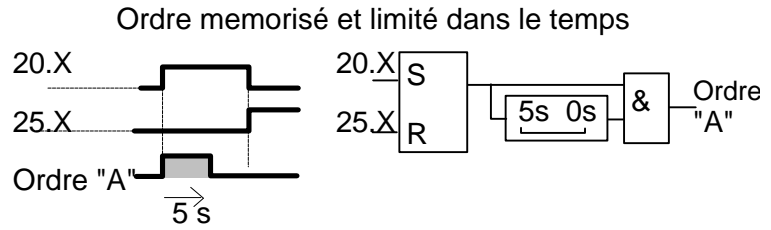
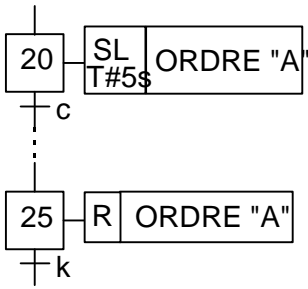
I-2-f) Convention IL

La programmation en IL (Instruction List) peut être directement insérée dans un bloc d'action, selon la syntaxe suivante :

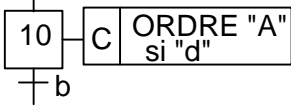
NORME 1131-3	ISaGRAF®
<pre> ACTION (P) : (* ou N *) <instruction> <instruction> END_ACTION; </pre>	<pre> ACTION (P) : (* ou N *) #info=IL <instruction> <instruction> #endinfo END_ACTION; </pre>
<p>(* Programme SFC et action programmée en IL *)</p> <pre> 1 --- ACTION (P) : LD False ST led1 ST led2 END_ACTION; </pre>	<p>(* Programme SFC et action programmée en IL *)</p> <pre> 1 --- ACTION (P) : #info=IL LD False ST led1 ST led2 #endinfo END_ACTION; </pre>

I-3) Exemples détaillés de qualificatifs d'action

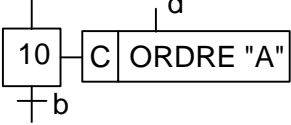




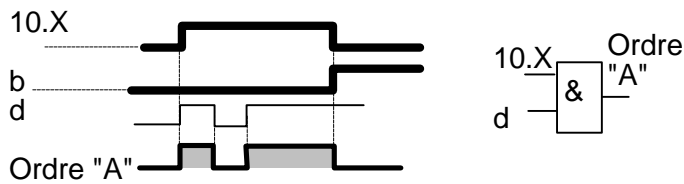
Forme 1



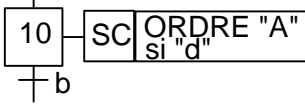
Forme 2



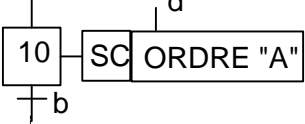
Ordre conditionnel



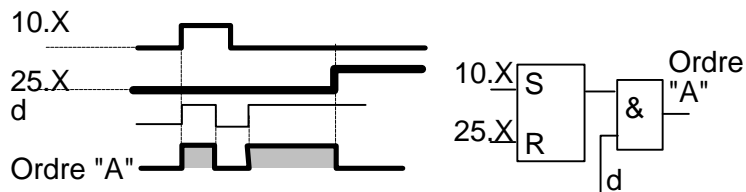
Forme 1



Forme 2



Ordre mémorisé conditionnel



Drapeau d'étape. A une étape repérée par *** on associe une variable d'étape noté ***.X
Temps écoulé pour une étape : ***.T

On notera l'équivalence des deux notations ci-dessous :

L	ACTION_1	DN1
t#10s		
P	ACTION_2	
N	ACTION_3	

```
STEP S8:
  ACTION_1(L,t#10s,DN1);
  ACTION_2(P);
  ACTION_3(N);
END_STEP
```

II) Conditions attachées aux transitions

A chaque transition est attachée une expression booléenne, qui conditionne le franchissement de la transition. Les conditions sont généralement décrites en langage ST. C'est le niveau 2 de la transition. Mais d'autres conventions d'écriture sont autorisées :

II-1) Convention ST

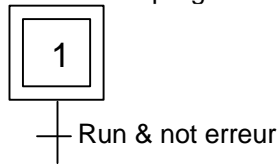
Vous pouvez utiliser le langage ST (Structured Text) pour décrire la condition attachée à une transition. L'expression doit être du type booléen et doit être terminée par un point virgule, selon une des syntaxes suivantes :

< expression_booleenne > ;	TRANSITION FROM XXX TO XXX < expression_booleenne > ; END_TRANSITION
----------------------------	--

XXX désigne une étape ou plusieurs étapes alors entre parenthèses.

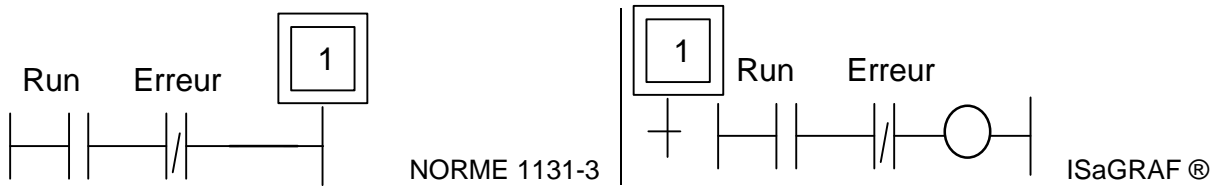
L'expression peut être une valeur constante TRUE ou FALSE, une variable booléenne interne ou d'entrée, ou une combinaison de variables représentant une grandeur booléenne. Voici un exemple de transition programmée en ST :

(* Programme SFC avec conditions programmées en ST *)



II-2) Convention LD

Le langage LD (schéma à relais) peut être utilisé pour décrire une condition associée à une transition. Le schéma est alors composé d'un seul échelon contenant un seul relais. La valeur du relais représente la valeur de la transition. Voici un exemple de transition programmée en LD:



II-3) Convention IL

Le langage IL (Instruction List) peut directement être utilisé pour programmer la condition d'une transition, selon la syntaxe suivante :

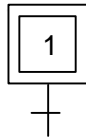
NORME 1131-3	ISaGRAF ®
TRANSITION FROM XXX TO XXX <instruction> <instruction> END_TRANSITION	#info=IL <instruction> <instruction> #endinfo

La valeur contenue dans le résultat courant (registre IL) à la fin de la séquence programmée indique le résultat de la condition attachée à la transition :

résultat courant = 0	condition = FALSE
résultat courant <> 0	condition = TRUE

Voici un exemple de condition programmée en IL :

(* Programme SFC avec transitions programmées en IL *)



TRANSITION FROM 1 TO 2:
LD Run
&N Erreur
END_TRANSITION

III) Exercices

1°) On désire gérer un feu de carrefour simple à 2 voies. On a une entrée jour nuit, et 6 sorties (2 x vert, orange, rouge) et on ne dispose pas de bit spécial type SY6 (1 Hz). Réaliser un GRAFCET qui réalise ce problème.

S'il est vrai qu'un programme SFC sert en général à décrire une commande d'une machine de production, la norme 1131-3 peut faire évoluer les choses. Les cibles visées par les programmes vont s'étendre de l'automate programmable aux microcontrôleurs. ISaGRAF ® propose un atelier logiciel pouvant générer du C. Les parties opératives seront alors des circuits électroniques.

2°) On peut reprendre le problème du tachymètre. Le schéma de principe est donné dans la page suivante. On suppose que l'on communique avec la partie opérative par :

- l'entrée capteur qui donne les impulsions venant du photocapteur,
- 8 sorties (1 digits).

a) L'affichage se faisant sur 4 digits il faudra multiplexer (voir figure ci-après). On suppose comme en TP que l'on a une roue percée de 60 trous pour un tour et qu'ainsi il suffit de compter le nombre de trous qui passent en une seconde et d'afficher le résultat pour avoir la vitesse en tour par minutes. Le multiplexage de l'affichage se fait toutes les 20 ms. Le compteur est supposé respecter la norme 1131, c'est à dire qu'il est actif sur niveau (et non sur front) et qu'il compte en entier. Utiliser donc la transformation INT_TO_BCD de la norme pour l'affichage. On demande d'écrire un programme SFC décrivant le séquençement de ce tachymètre.

b) Ajouter ensuite la possibilité de déclencher le relais si la vitesse dépasse une valeur pouvant être programmée. La programmation de cette vitesse se fait en appuyant sur SW1 et SW3 pour incrémenter et sur SW2 et SW3 pour décrémenter. L'appui sur SW3 seul permet uniquement de consulter la valeur de cette vitesse maximale. L'incrémentation (ou la décrémentation) se fait d'abord lentement puis de plus en plus vite au fur et à mesure que le temps pendant lequel on appuie augmente.

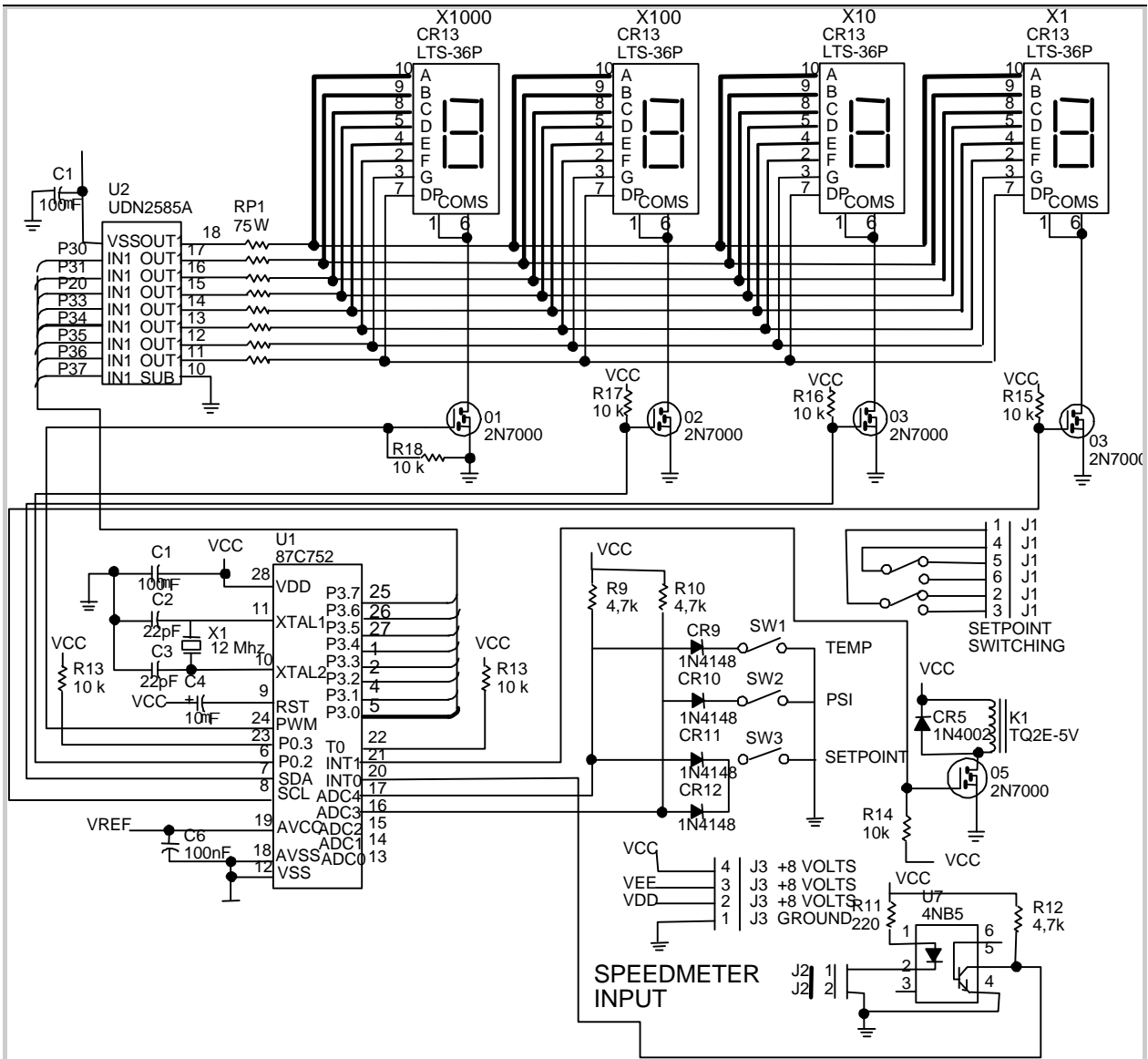


TABLE DES MATIERES

Cours 1 - La numération	1
I - Définitions	1
I-1) Expression générale.....	1
I-2) Rappels : la numération décimale.	1
II - Système de numération binaire.	1
II-1) Expression	1
II-2) Conversion Binaire-Décimal.....	1
II-3) Conversion Décimal-Binaire.....	1
II-4) Cas des nombres fractionnaires.	2
II-5) Précision fractionnaire.....	2
II-6) Utilisation pour la longueur de conversion décimale-binaire.	2
III - Autres systèmes de numérations.....	3
III-1) Système de numération octale.	3
III-2) Système de numération hexadécimale.....	3
III-3) Conversions entre systèmes de numération (2, 8 ou 16).	5
IV - Exercices.....	6
Cours 2 - Représentation des nombres.	7
I - Représentation des entiers positifs.	7
II - Représentation des entiers relatifs.....	7
II-1) représentation par bit de signe et valeur absolue.....	7
II-2) Représentation par le complément restreint de N (complément à 1).....	8
II-3) Représentation par le complément vrai de N (complément à deux).....	8
III - Représentation des nombres fractionnaires.....	10
III - 1) Virgule placée à un rang fixe quelconque.....	10
III - 2) virgule située à droite du dernier rang.	10
III-3) Virgule située à droite du bit de signe.....	11
III-4) Norme ANSI IEEE standard 754 pour la représentation des réels.....	11
IV - Exercices.....	11
Cours 3 - Les Codes	13
I - Définitions.	13
II - Différents types de codes.	13
II-1) Codes arithmétiques.	14
II-2) Codes de position : (réfléchis ou reflex).	14
II-3) Codes détecteurs.	15
II-4) Codes alphanumériques :	16
III - Exercice.....	17
Cours 4 : Le langage ABEL.....	18
I - Le langage ABEL (spécifications)	18
I-1) Caractères valides	18
I-2) Identificateurs.	18
I-3) Constantes.....	18
I-4) Blocs.....	18
I-5) Commentaires.	18
I-6) Nombres.....	18
I-7) Chaînes de caractères.	19
I-8) Opérateurs expressions et équations.....	19
I-9) Les ensembles.....	19
I-10) La structure de programme.	19
I-11) Les points extensions.....	20
I-12) Les macros.	21
II) Applications.....	21
II-1) Passage table de vérité -> équations	22
II-2) Simulation.....	22
II-3) Introduction des équations logiques.....	22
II-4) Plusieurs sorties.....	22
II-5) Méthode SI-ALORS	22
TD1 - Logique et algèbre de Boole.....	23
I - Définitions.....	23

II - Représentation des fonctions logiques.....	23
III - Fonctions élémentaires.	24
III-1) Fonction d'une variable	24
III-2) Fonction de deux variables : $f(a,b)$	24
IV - Exercice : les 16 fonctions $F(a,b)$	26
TD 2 - Les fonctions Booléennes.....	27
I - Définitions.	27
II - Les représentations.	27
III - Exercices.	28
TD 3 - Simplification des fonctions logiques.....	29
I - Tableaux de Karnaugh.	29
II - Méthodes algébriques.	30
II-1) Emploi des théorèmes de l'algèbre de Boole :.....	30
II-2) Méthode du consensus.....	31
III - Exercices.	31
TD 4 - Réalisation des circuits.....	32
I - Synthèse avec la structure ET/OU.....	32
II - Synthèse à ET-Non.	32
III - Portes ET-Non limitées par le nombre d'entrée.....	33
IV - Synthèse avec la structure OU/ET.	33
V - Synthèse à OU-Non.	33
VI - Portes OU-Non limitées par le nombre d'entrée.	34
VII - Optimisations.	34
VIII - Exercices	35
TD 5 - Méthode du SI-ALORS.....	36
I - Présentation du tableau SI-ALORS	36
I-1) Table de vérité.	36
I-2) Tableau SI-ALORS	36
II - Exemples.	36
III - Tables SI-ALORS et le langage ABEL.....	37
IV - Exercices.	38
Devoir surveillé n°1 d'informatique industrielle.....	40
Cours 5 - Circuits de transcodage.	42
I - Réalisation d'un circuit de transcodage.....	42
II - Circuits existants.	42
Cours 6 - Multiplexage - Demultiplexage.....	44
I - Définitions.	44
II - Application particulières.....	45
Cours 7 - Paramètres électriques et temporels.	46
I - Familles et sous familles.	46
I-1) TTL.....	46
I-2) ECL	46
I-3) CMOS.....	46
II - Paramètres électriques.....	46
II-1) Niveaux d'entrée et de sortie.....	46
II-2) Immunité aux bruits	47
II-3) Courants de sortie et d'entrée.....	47
II-4) Courant de court-circuit.....	47
III - Paramètres temporels.	48
IV - Exercices.	48
Cours 8 - Portes collecteur ouvert, trois états, trigger de schmitt.	49
I - Portes à sortie collecteur ouvert.	49
II - Porte à sortie 3 états.	49
III - Portes à entrées trigger de schmitt.	50
TD 6 - La fonction mémoire.	51
I - Mémoire RS.	51
I-1) Principe de fonctionnement.....	51
I-2) Exemples de bistable RS.	51
I-3) Méthodes d'études.	51
II - La mémoire D.	51

III - La mémoire RST.....	52
IV - Exercices.....	52
TD 7 - Les bascules.....	53
I - Fonction bascule.....	53
II - Comptage.....	53
III - Les différentes bascules.....	53
IV - Bascule JK.....	53
V - Bascule T.....	54
VI - Bascule D.....	54
VII - Exercices.....	55
TD 8 - Les registres.....	57
I - Registres.....	57
I-1) Définitions.....	57
I-2) Structures des registres.....	57
I-3) Ecriture d'un mot dans un registre.....	57
I-4) Lecture d'un mot contenu dans un registre.....	58
II - Exercices.....	59
TD 9 - Les compteurs synchrones.....	62
I - Analyse des machines pulsées à franchissement synchrone.....	62
I-1) Analyse d'un compteur synchrone de type registre en anneau.....	62
I-2) Analyse d'un système ayant plusieurs cycles.....	62
I-3) Générateur de chiffres pseudo-aléatoires.....	63
I-4) Méthode d'analyse d'un système quelconque.....	63
I-5) Montage avec JK.....	64
II - Synthèse des compteurs synchrones.....	64
III - Exercices.....	66
IV - Limitation en fréquence des compteurs synchrones.....	66
V - Initialisation d'un compteur.....	67
VI - Forçage asynchrone des compteurs synchrones (à rétroaction).....	67
VII - Compteurs et le langage ABEL.....	68
TD 10 - Compteurs asynchrones.....	70
I - Analyse de circuits existants.....	70
I-1) Présentation générale : le compteur binaire.....	70
I-2) Le compteur 7493.....	70
I-3) Le compteur 7490.....	70
II - Exercice.....	71
III - Synthèse.....	71
Devoir Surveillé d'informatique industrielle n°2.....	72
TD11 - Le GRAFCET (outil de description).....	76
I - Système automatisé.....	76
I-1) Généralités.....	76
I-2) La démarche de conception d'un système automatisé de production.....	76
II - Description par GRAFCET.....	76
III - Le GRAFCET (règles d'établissement).....	79
III-1) Etapes.....	79
III-2) Transitions.....	80
III-3) Graphe des états.....	81
IV - EXERCICES.....	82
TD12 - GRAFCET (synthèse matérielle non programmée).....	85
I - Voir le GRAFCET comme une description de machine séquentielle.....	85
II - Généralités.....	85
III - Méthode asynchrone.....	85
IV - Méthodes synchrones.....	86
IV-1) Méthode d'activation-désactivation synchrone.....	86
IV-2) Méthode optimisée en nombre de bascules.....	87
V - Exercices.....	87
TD 13 - Logique programmable et ABEL.....	89
I - Présentation de la logique programmable.....	89
I-1) Les conventions de représentation des PALs.....	89
I-2) Notre GAL 20V8.....	89

II - Exercices	90
IV - COMPLEMENT : PROGRAMMATION D'UN GRAFCET EN ABEL	94
TD 14 - GRAFCET : Synthèse programmée.	95
I - Bien comprendre l'architecture.	95
I-1) Mémoires.	95
I-2) Analyse.	95
II - Synthèse (mémoire plus registre).	96
III - Synthèse compteur plus mémoire.	97
III-1) Séquenceur à enchaînement séquentiel.....	97
III-2) Séquenceurs à enchaînement conditionnel	98
III-3) Séquenceur avec branchement inconditionnel	99
III-4) Séquenceur avec branchement conditionnel	100
IV - Exercice.....	100
DS Informatique Industrielle n°3	101
TD 15 - Les opérations arithmétiques.....	105
I - Addition binaire.	105
I-1) Principe :.....	105
I- 2) Addition parallèle à retenue en cascade :	106
II - La soustraction.	107
II-1) Principe (rappel).....	107
II-2) Soustraction parallèle par complément vrai.....	107
II-3) Addition/soustraction en complément vrai.	108
III - Exercices - Additionneurs / Soustracteurs.....	108
TD 16 - Comparaison et multiplication.et calculette.....	110
I - Comparaison.....	110
II - Multiplication	110
II-1) Table de multiplication binaire (à un bit).....	111
II-2) Réalisation d'un multiplieur de deux nombres 3 bits	111
II-3) Circuit intégré.....	111
III - Architectures d'une calculette simple	111
IV - Exercices	112
TD 17 - CEI 1131-3 : Le langage IL et LD	113
I - Architecture mémoire plus compteur plus UAL	113
II - Langage IL	113
II-1) La norme 1131-3 (Extraits choisis).....	113
II -2) Exemple du TSX 37 (Telemecanique).....	118
II - 3) Programmation de GRAFCETs en langage IL.....	118
IV - Exercices	119
TD 18 - CEI 1131-3 Langage SFC et ST	122
I) Programmation des actions	122
I-1) Généralités sur les actions	122
I-2) Les différentes actions	122
I-3) Exemples détaillés de qualificatifs d'action.....	124
II) Conditions attachées aux transitions.....	126
II-1) Convention ST	127
II-2) Convention LD	127
II-3) Convention IL.....	127
III) Exercices	128
TABLE DES MATIERES	130