Introduction aux circuits logiques

DEUXIÈME ÉDITION

Letocha

INTRODUCTION AUX CIRCUITS LOGIQUES

DEUXIÈME ÉDITION

JEAN LETOCHA
Collège Ahuntsic

LÉON COLLET
Conseiller technique et linguistique

McGraw-Hill, Éditeurs

Montréal • Toronto • New York • Saint-Louis • San Francisco • Auckland • Bogotá • Guatemala • Hambourg • Johannesburg • Lisbonne • Londres • Madrid • Mexico • New Delhi • Panama • Paris • San Juan • São Paulo • Singapour • Sydney • Tokyo

La société TEXAS INSTRUMENTS INC. a donné l'autorisation de reproduire les configurations des boîtiers des circuits intégrés ainsi que les diagrammes fonctionnels.
L'Agence nationale pour le développement de la production automatisée (ADEPA) a donné son accord pour la reproduction de la brochure GRAFCET à l'annexe A.
Ces deux sociétés sont vivement remerciées de leur amabilité.
Nous remercions également monsieur Richard Lemoyne du collège Édouard-Montpetit pour avoir préparé l'index des schémas de circuits intégrés à la fin du volume.

INTRODUCTION AUX CIRCUITS LOGIQUES. DEUXIÈME ÉDITION
Copyright © 1982, 1985
McGraw-Hill, Éditeurs.
Tous droits réservés. On ne peut reproduire, enregistrer, ni diffuser aucune partie du présent ouvrage, sous quelque forme ou par quelque procédé que ce soit, électronique, mécanique, photographique, sonore, magnétique ou autre, sans avoir obtenu au préalable l'autorisation écrite de l'éditeur.

Dépôt légal 1er trimestre 1985
Bibliothèque nationale du Québec
Imprimé et relié au Canada

2 3 4 5 6 7 8 9 0 BI85 0 9 8 7 6
ISBN 0-07-548985-6

Avant-propos

Ce livre est un ouvrage d'initiation à l'algèbre de Boole et à ses applications aux circuits à deux états appelés circuits logiques. La plupart des problèmes envisagés traitent des circuits combinatoires. Les circuits à mémoires ou circuits séquentiels sont exposés dans le dernier chapitre.

Ce livre est surtout conçu pour le cours intitulé «Circuits logiques». De plus, il permet aux étudiants d'aborder des notions exposées au cours intitulé «Techniques numériques». Il comprend en cinq chapitres les grands sujets à étudier dans un cours d'initiation. Tous les points traités dans un chapitre ne sont pas obligatoirement des préalables à un chapitre ultérieur. À chaque chapitre le professeur tiendra compte du temps disponible, de la vitesse d'apprentissage des étudiants et de l'indispensable possession de certaines notions pour choisir les sections à étudier immédiatement ou plus tard. On peut, par exemple, ignorer le premier chapitre quitte à y revenir lorsque ses notions sont nécessaires à la résolution des problèmes d'un chapitre ultérieur. Ainsi, au moment d'étudier le demi-additionneur ou les circuits de passage d'un code à l'autre, on reviendra au premier chapitre pour expliquer les notions indispensables à leur compréhension. Cet ouvrage peut donc être utilisé pour les cours réguliers du jour et du soir ainsi que pour le recyclage des techniciens.

Puisse ce livre de facture moderne, aux exemples faisant appel à la technologie de pointe, servir de préambule à l'étude, indispensable de nos jours, des microprocesseurs.

Table des matières

1

Systèmes de numération

1-1 OBJECTIFS

1. Savoir définir la base d'un système de numération.
2. Savoir définir le rang, ou poids, d'un chiffre.
3. Savoir représenter un nombre de base b quelconque sous forme polynomiale.
4. Savoir déterminer la valeur décimale d'un nombre de base b quelconque.
5. Savoir convertir un nombre de base décimale en un nombre de base b quelconque (selon les deux procédés décrits).
6. Savoir convertir un nombre binaire en un nombre octal ou en un nombre hexadécimal et vice versa.
7. Savoir effectuer les quatre opérations ($+$, $-$, $\times$, $\div$) directement dans le système binaire naturel.
8. Savoir complémenter à 1 et à 2 un nombre binaire et savoir appliquer cette représentation à la soustraction.
9. Savoir écrire un nombre binaire sous forme normalisée.
10. Savoir comment un calculateur additionne et soustrait automatiquement.
11. Savoir coder un nombre décimal et un nombre binaire en Gray et vice versa.
12. Savoir coder un nombre décimal en BCD et vice versa.
13. Savoir coder un nombre décimal ou BCD en code «plus trois» et vice versa.
14. Connaître les principaux codes détecteurs et correcteurs d'erreurs et savoir comment la machine détecte et corrige une erreur.
15. Savoir coder et décoder une carte perforée selon le code Hollerith.
16. Savoir coder et décoder un ruban perforé selon le code ASCII.
17. Savoir ce qu'on entend par impulsions série et impulsions parallèle, leur avantage et leur inconvénient respectifs et savoir déterminer le nombre requis de fils pour émettre, dans ces deux modes, un nombre binaire donné.

1-2 INTRODUCTION

Ce chapitre expose la transition entre le codage et le fonctionnement intrinsèque d'un calculateur électronique automatique ou ordinateur.

Nous donnerons des exemples de codage pour le calcul numérique et nous vérifierons que l'algorithme d'une opération ne dépend pas du système de numération choisi.

Nous n'exposerons pas la théorie des quatre opérations élémentaires (addition, soustraction, multiplication, division) mais nous établirons un parallèle entre les techniques de ces opérations dans les systèmes de numération binaire et décimal.

Nous verrons de plus les premières notions de codage chiffré. Notions essentielles, car dans un ordinateur, même les caractères de l'alphabet (a, b, c, . . . , z) sont codés sous forme numérique binaire. Il faut donc se familiariser avec le calcul binaire et le codage.

L'ordinateur ne connaît que les chiffres, mais on peut, par son intermédiaire, commander un téléviseur couleur, écrire une partition, tracer des schémas ou des esquisses, etc.

1-3 BASE D'UN SYSTÈME DE NUMÉRATION

1-3-1 FORME POLYNOMIALE

On peut décomposer tout nombre N en fonction des puissances entières de la *base** de son *système de numération*. Considérons, par exemple, le nombre *décimal* (base 10) 93452. On notera en indice la base du système de numération dans lequel le nombre N envisagé est écrit (sauf cas particuliers, on négligera de préciser les bases 2 et 10). On aura:

$$(93452)_{10} \equiv 9 \times 10^4 + 3 \times 10^3 + 4 \times 10^2 + 5 \times 10^1 + 2 \times 10^0$$

Par définition, à notre échelle, tout nombre élevé à la puissance 0 égale 1: $a^0 = 1$ quel que soit a.

Dans le système décimal nous disposons des dix symboles (appelés *chiffres*) notés 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Dans un nombre quelconque le chiffre de droite (2 dans l'exemple ci dessus) s'appelle le chiffre de *poids faible*, celui de gauche (9 dans notre exemple) s'appelle le chiffre de *poids fort*.

Exemple 1 Écrire $N = (27674)_{10}$ sous forme polynomiale et déterminer les chiffres de poids fort et faible.

Solution Nous aurons:

$$(27674)_{10} \equiv 2 \times 10^4 + 7 \times 10^3 + 6 \times 10^2 + 7 \times 10^1 + 4 \times 10^0$$

chiffre de poids fort: 2, chiffre de poids faible: 4.

*La base d'un système de numération est le nombre de chiffres différents qu'utilise ce système de numération.

On peut généraliser cette notion et écrire sous une forme plus abstraite tout nombre décimal N de n + 1 chiffres.

On aura:

$$N \equiv \sum_{i=0}^{i=n} a_i \times 10^i$$

où a_i est un chiffre tel que $0 \leqslant a_i \leqslant 9$, les i sont des entiers $\geqslant 0$ et n est l'exposant de 10 du chiffre de poids fort.

Le signe $\sum$, *sigma*, est l'opérateur sommation de tous les monômes à sa droite pour i variant de 0 à n. Le signe $\equiv$ signifie *identique à*.

1-3-2 RANG D'UN CHIFFRE

Le *rang* d'un chiffre d'un nombre de base b quelconque est égal à l'exposant de la base associée à ce chiffre dans la représentation polynomiale du nombre considéré. Soit par exemple, $N = (87672)_{10}$, alors: 2 est de rang 0, 6 est de rang 2, 8 est de rang 4.

Le rang des chiffres croît de la droite vers la gauche. Le rang du chiffre de droite est nul par définition.

Exemple 1 Soit $N = (23456)_{10}$. Déterminer:

a) le rang du chiffre 5 **(rép. 1)**

b) le rang du chiffre 3 **(rép. 3)**

REMARQUE: Au lieu de rang on peut employer «poids».

1-3-3 REPRÉSENTATION POLYNOMIALE D'UN NOMBRE N DE BASE b QUELCONQUE

Nous avons vu à la section 1-3-1 la décomposition de tout nombre décimal N en fonction de ses puissances entières de 10, soit:

$$N \equiv \sum_{i=0}^{i=n} a_i \times 10^i$$

où a_i est un chiffre tel que $0 \leqslant a_i \leqslant 9$, les i sont des entiers $\geqslant 0$ et n est l'exposant de 10 du chiffre de poids fort.

Il importe de noter que les symboles notés de 0 à 9 forment un ensemble ordonné. On aurait dû écrire en toute rigueur mathématique:

$a_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ qui se lit:

a_i appartient ($\in$) à l'ensemble ordonné $\{0, 1, 2, \ldots, 9\}$ ou encore a_i est un élément de l'ensemble ordonné $\{0, 1, 2, \ldots, 9\}$.

Dans le cas d'une base b quelconque on respectera la même notation et les mêmes conventions. Tout nombre N de base b sera donc décomposable en fonction des puissances entières de b.

On aura donc:

$$N \equiv \sum_{i=0}^{i=n} a_i \times b^i$$

où $a_i \in \{0, 1, 2, \ldots, (b-1)\}$, les i sont des entiers $\geqslant 0$ et n est l'exposant de b du chiffre de poids fort.

Exemple 1 Dans le système à base 6, on aura:

$$N \equiv \sum_{i=0}^{i=n} a_i \times 6^i$$

où $a_i \in \{0, 1, 2, 3, 4, 5\}$, les i sont des entiers $\geqslant 0$ et n est l'exposant de 6 du chiffre de poids fort.

soit,

$$(54321)_6 \equiv 5 \times 6^4 + 4 \times 6^3 + 3 \times 6^2 + 2 \times 6^1 + 1 \times 6^0$$

Exemple 2 Dans le système à base 9, on aura:

$$N \equiv \sum_{i=0}^{i=n} a_i \times 9^i$$

où $a_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$, les i sont des entiers $\geqslant 0$ et n est l'exposant de 9 du chiffre de poids fort.

soit,

$$(87836)_9 \equiv 8 \times 9^4 + 7 \times 9^3 + 8 \times 9^2 + 3 \times 9^1 + 6 \times 9^0$$

Exemple 3 Comment écrire un nombre de base 12?

Solution

$$N \equiv \sum_{i=0}^{i=n} a_i \times 12^i$$

où $a_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \ldots\}$, les i sont des entiers $\geqslant 0$ et n est l'exposant de 12 du chiffre de poids fort.

Il nous manque deux chiffres. Selon la convention nord-américaine, nous prendrons les premières lettres A, B de l'alphabet.

On aura donc dans ce cas l'ensemble des chiffres suivants:

$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B\}$

Ce système de numération est dit *duodécimal*.

soit,

$$(9A73B)_{12} \equiv 9 \times 12^4 + A \times 12^3 + 7 \times 12^2 + 3 \times 12^1 + B \times 12^0$$

Exemple 4 Si b = 2, le système de numération est appelé *binaire* et l'on aura:

$$N \equiv \sum_{i=0}^{i=n} a_i \times 2^i$$

où $a_i \in \{0,1\}$, les i sont des entiers $\geqslant 0$ et n est l'exposant de 2 du chiffre de poids fort.

soit,

$$(101101)_2 \equiv 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

1-3-4 VALEUR DÉCIMALE D'UN NOMBRE N DE BASE b QUELCONQUE

La *valeur décimale* d'un nombre N de base b s'obtient par sa forme polynomiale vue au paragraphe précédent.

Exemple 1 Déterminer la valeur décimale de N = (101101).

Solution Nous aurons:

$$\begin{aligned}(101101)_2 &\equiv 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\ &\equiv 1 \times 32 + 0 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 \\ &\equiv 32 + 8 + 4 + 1 \\ &= (45)_{10}\end{aligned}$$

Exemple 2 Déterminer la valeur décimale du nombre *octal* (base 8) $N = (6734)_8$.

Solution Nous aurons:

$$\begin{aligned}(6734)_8 &\equiv 6 \times 8^3 + 7 \times 8^2 + 3 \times 8^1 + 4 \times 8^0 \\ &\equiv 6 \times 512 + 7 \times 64 + 3 \times 8 + 4 \times 1 \\ &\equiv 3072 + 448 + 24 + 4 \\ &= (3458)_{10}\end{aligned}$$

Exemple 3 Déterminer la valeur décimale du nombre *hexadécimal* (base 16) $(A732)_{16}$.

Solution Nous aurons:

$$\begin{aligned}(A732)_{16} &\equiv A \times 16^3 + 7 \times 16^2 + 3 \times 16^1 + 2 \times 16^0 \\ &\equiv 10 \times 4096 + 7 \times 256 + 3 \times 16 + 2 \\ &\equiv 40960 + 1792 + 48 + 2 \\ &= (42802)_{10}\end{aligned}$$

1-4 CHANGEMENT DE BASE

1-4-1 PREMIER PROCÉDÉ DE CONVERSION D'UN NOMBRE DE BASE DÉCIMALE EN UN NOMBRE DE BASE b QUELCONQUE

Exemple 1 Convertir le nombre $N = (39487)_{10}$ en nombre octal.

Solution Nous avons vu à la section 1-3-3 que le nombre N cherché s'écrit:

$$N \equiv \sum_{i=0}^{i=n} a_i \times 8^i$$

où $a_i \in \{0, 1, 2, 3, 4, 5, 6, 7\}$, les i sont des entiers $\geqslant 0$ et n est l'exposant de 8 du chiffre de poids fort.

Le problème revient à déterminer les valeurs de a_i. Pour cela, appliquer l'*algorithme* suivant: chercher la plus grande puissance entière de 8 contenue dans $N = (39487)_{10}$, retrancher cette quantité de $(39487)_{10}$, considérer maintenant le reste obtenu et recommencer ce processus.

Il faut donc connaître les différentes puissances entières de 8. Elles figurent dans la table ci-dessous.

i	8^i
0	1
1	8
2	64
3	512
4	4096
5	32768

On aura successivement:

$$
\begin{array}{rl}
 & 39487 \\
- & 32768 \quad \text{----------} \rightarrow 8^5 \\
 & 06719 \\
- & 4096 \quad \text{----------} \rightarrow 8^4 \\
 & 2623 \\
- & 512 \quad \text{----------} \rightarrow 8^3 \\
 & 2111 \\
- & 512 \quad \text{----------} \rightarrow 8^3 \\
 & 1599 \\
- & 512 \quad \text{----------} \rightarrow 8^3 \\
 & 1087 \\
- & 512 \quad \text{----------} \rightarrow 8^3 \\
 & 575 \\
- & 512 \quad \text{----------} \rightarrow 8^3 \\
 & 063 \\
7 \times 8 = & 56 \quad \text{----------} \rightarrow 7 \times 8^1 \\
 & 07 \quad \text{----------} \rightarrow 7 \times 8^0
\end{array}
$$

Donc:

$$N = (39487)_{10} \equiv 1 \times 8^5 + 1 \times 8^4 + 5 \times 8^3 + 7 \times 8^1 + 7 \times 8^0$$

On voit que le terme 8^2 est absent, d'où $a_2 = 0$, d'après notre relation.

On a donc:

$$N = (39487)_{10} = (115077)_8$$

Exemple 2 Convertir $N = (47375)_{10}$ en binaire.

Solution Comme précédemment pour 8, dressons une table des puissances entières de 2 et retranchons chaque fois la plus grande puissance entière de 2 possible.

i	2^i
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024
11	2048
12	4096
13	8192
14	16384
15	32768

$$
\begin{array}{rr}
 & 47375 \\
- & 32768 \quad \text{----------} \rightarrow 2^{15} \\
\hline
 & 14607 \\
- & 8192 \quad \text{----------} \rightarrow 2^{13} \\
\hline
 & 6415 \\
- & 4096 \quad \text{----------} \rightarrow 2^{12} \\
\hline
 & 2319 \\
- & 2048 \quad \text{----------} \rightarrow 2^{11} \\
\hline
 & 0271 \\
- & 256 \quad \text{----------} \rightarrow 2^{8} \\
\hline
 & 015 \\
- & 8 \quad \text{----------} \rightarrow 2^{3} \\
\hline
 & 7 \\
- & 4 \quad \text{----------} \rightarrow 2^{2} \\
\hline
 & 3 \\
- & 2 \quad \text{----------} \rightarrow 2^{1} \\
\hline
 & 1 \\
- & 1 \quad \text{----------} \rightarrow 2^{0} \\
\hline
 & 0
\end{array}
$$

Et l'on a:

$$N = (47375)_{10} \equiv 2^{15} + 2^{13} + 2^{12} + 2^{11} + 2^{8} + 2^{3} + 2^{2} + 2^{1} + 2^{0}$$

On voit que les termes $2^{14}, 2^{10}, 2^{9}, 2^{7}, 2^{6}, 2^{5}, 2^{4}$ sont absents. Nous en concluons donc que $a_{14} = a_{10} = a_9 = a_7 = a_6 = a_5 = a_4 = 0$.

Donc, $N = (47375)_{10} = (1011100100001111)_2$

RÉSUMÉ Dans ce premier algorithme on applique directement la formule:

$$N \equiv \sum_{i=0}^{i=n} a_i \, b^i$$

où $a_i \in \{0, 1, 2, \ldots, (b\text{-}1)\}$, les i sont des entiers $\geqslant 0$ et n est l'exposant de b du chiffre de poids fort.

On dresse une table donnant les différentes puissances entières de la base b du système de numération dans lequel on veut convertir le nombre décimal.

Au nombre décimal donné, on retranche la plus grande puissance entière de b possible. Cette puissance définit le rang du chiffre dans la

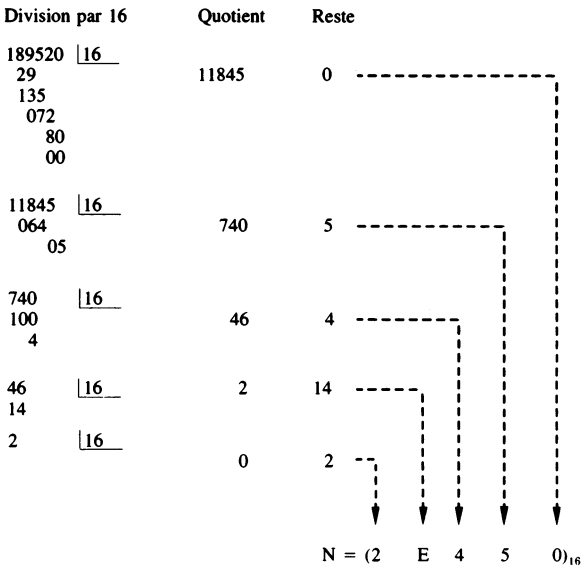
représentation en base b du nombre. Le nombre de fois (< b) qu'on retranche cette puissance définit le chiffre de ce rang. (En binaire, on ne peut avoir que 1 ou 0, la valeur du chiffre à écrire est donc immédiate).

1-4-2 DEUXIÈME PROCÉDÉ DE CONVERSION D'UN NOMBRE DE BASE DÉCIMALE EN UN NOMBRE DE BASE b QUELCONQUE

Cette méthode est simple et plus rapide que la précédente. Il est donc conseillé de l'utiliser dans tous les problèmes de conversion. Nous l'illustrerons à l'aide de deux exemples.

Exemple 1 Convertir le nombre $N = (189520)_{10}$ en hexadécimal.

Solution



Donc, $N = (189520)_{10} = (2E450)_{16}$

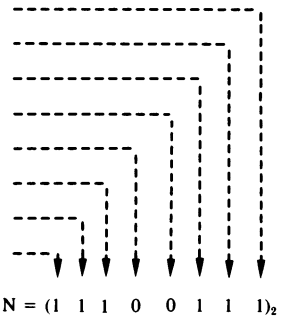
Vérification:

$$\begin{aligned} N &\equiv 2 \times 16^4 + 14 \times 16^3 + 4 \times 16^2 + 5 \times 16^1 + 0 \times 16^0 \\ &\equiv 2 \times 65536 + 14 \times 4096 + 4 \times 256 + 5 \times 16 + 0 \\ &\equiv 131072 + 57344 + 1024 + 80 \\ &= (189520)_{10} \end{aligned}$$

Exemple 2 Convertir le nombre $N = (231)_{10}$ en binaire.

Solution

Division par 2	Quotient	Reste
231 | 2		
115 | 2	115	1
57 | 2	57	1
28 | 2	28	1
14 | 2	14	0
7 | 2	7	0
3 | 2	3	1
1 | 2	1	1
	0	1



Donc, $N = (231)_{10} = (11100111)_2$

Vérification:

$$\begin{aligned} N &\equiv 2^7 + 2^6 + 2^5 + 2^2 + 2^1 + 2^0 \\ &\equiv 128 + 64 + 32 + 4 + 2 + 1 \\ &= (231)_{10} \end{aligned}$$

RÉSUMÉ Cet algorithme consiste à diviser le nombre à convertir par la base du nouveau système et à conserver le reste. On répète ce processus en considérant chaque fois le quotient obtenu. On écrit ensuite tous les restes à partir de la fin et de gauche à droite, en les convertissant en lettres s'il y a lieu.

JUSTIFICATION DE LA MÉTHODE Soit N le nombre à convertir dans le système de numération de base b.

$N = q_0 b + r_0$ $\quad$ $q_0, q_1, q_2, \ldots q_{n-1}$: quotients
$r_0, r_1, r_2, \ldots, r_n$: restes

$q_0 = q_1 b + r_1$

$q_1 = q_2 b + r_2$

$\vdots$

$q_{n-3} = q_{n-2} b + r_{n-2}$

$q_{n-2} = q_{n-1} b + r_{n-1}$

$q_{n-1} = ob + r_n$

D'où:

$$q_{n-2} = r_n b + r_{n-1}$$
$$q_{n-3} = (r_n b + r_{n-1})\, b + r_{n-2}$$
$$= r_n b^2 + r_{n-1} b + r_{n-2}$$
$$\vdots$$
$$q_0 = r_n b^{n-1} + r_{n-1} b^{n-2} + \ldots + r_1$$

D'où:

$$N = r_n b^n + r_{n-1} n^{n-1} + \ldots + r_1 b + r_0 b^0$$

Les restes r_i sont bien les coefficients de la décomposition polynomiale, donc

$$N \equiv \sum_{i=0}^{i=n} r_i b^i$$

où $r_i \in \{0, 1, 2, \ldots, (b-1)\}$, les i sont des entiers $\geqslant 0$ et n est l'exposant de b du chiffre de poids fort.

1-4-3 NOMBRES FRACTIONNAIRES

Rappel sur les nombres fractionnaires de base 10

Nous savons qu'un nombre fractionnaire décimal N, par exemple $(0{,}8237421)_{10}$ peut se décomposer sous la forme:

$$N \equiv 8 \times 10^{-1} + 2 \times 10^{-2} + 3 \times 10^{-3} + 7 \times 10^{-4} + 4 \times 10^{-5} + 2 \times 10^{-6} + 1 \times 10^{-7}$$

Les rangs des termes seront: $-1, -2, -3, -4, -5, -6$ et -7. D'une façon générale, si un nombre fractionnaire décimal r/s est inférieur à 1, on peut le mettre sous la forme:

$$r/s \equiv \sum_{i=1}^{i=n} a_i\, 10^{-i}$$

où $a_i \in \{0, 1, 2, \ldots, 9\}$, les i sont des entiers > 0 et $-n$ est l'exposant de 10 du chiffre de poids faible.

Si le système envisagé est de base b, on adoptera les mêmes conventions; on écrira donc un nombre fractionnaire sous la forme:

$$r/s \equiv \sum_{i=1}^{i=n} a_i b^{-i}$$

où $a_i \in \{0, 1, 2, \ldots, (b-1)\}$, les i sont des entiers >0 et $-n$ est l'exposant de b du chiffre de poids faible.

Problème: Convertir un nombre fractionnaire inférieur à 1 de base b en décimal.

Exemple 1 Convertir $N = (0,1011001101)_2$ en décimal.

Solution Nous aurons:

$$N \equiv 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4} + 0 \times 2^{-5} + 0 \times 2^{-6} + 1 \times 2^{-7} + 1 \times 2^{-8} + 0 \times 2^{-9} + 1 \times 2^{-10}$$

Pour calculer ce nombre, il nous faut les valeurs décimales des puissances négatives de 2. Elles apparaissent dans la table suivante.

i	2^{-i}
1	0,5
2	0,25
3	0,125
4	0,0625
5	0,03125
6	0,015625
7	0,0078125
8	0,00390625
9	0,001953125
10	0,0009765625

la somme

$$\begin{array}{r} 0,5 \\ + \ 0,125 \\ + \ 0,0625 \\ + \ 0,0078125 \\ + \ 0,00390625 \\ + \ 0,0009765625 \\ \hline \end{array}$$

donne $N = (0,7001953125)_{10}$

Exemple 2 Convertir $N = (0,163)_8$ en décimal.

Solution Nous aurons:

$$N \equiv 1 \times 8^{-1} + 6 \times 8^{-2} + 3 \times 8^{-3}$$

i	8^{-i}
1	0,125
2	0,15625
3	0,001953125

la somme
$$\begin{array}{r} 0{,}125 \\ +\ 0{,}093750 \\ +\ 0{,}005859375 \\ \hline \end{array}$$
donne $N = (0{,}224609375)_{10}$

Problème: Convertir un nombre décimal fractionnaire en un nombre de base b.

Exemple 1 Convertir $N = (0{,}72145)_{10}$ en binaire.

Solution L'algorithme utilisé est analogue à celui de la deuxième méthode de conversion vue à la section 1-4-2, mais au lieu d'une division on aura une multiplication. Sa justification, de même nature, est laissée au lecteur. On aura donc successivement:

$0{,}72145 \times 2 = \boxed{1}\ ,\ 44290$
$0{,}44290 \times 2 = \boxed{0}\ ,\ 88580$
$0{,}88580 \times 2 = \boxed{1}\ ,\ 77160$
$0{,}77160 \times 2 = \boxed{1}\ ,\ 54320$
$0{,}54320 \times 2 = \boxed{1}\ ,\ 08640$
$0{,}08640 \times 2 = \boxed{0}\ ,\ 17280$
$0{,}17280 \times 2 = \boxed{0}\ ,\ 34560$
$0{,}34560 \times 2 = \boxed{0}\ ,\ 69120$
$0{,}69120 \times 2 = \boxed{1}\ ,\ 38240$

Donc, $N = (0{,}72145)_{10} = (0{,}101110001)_2$

Il suffit donc d'écrire de gauche à droite les nombres encadrés pris de haut en bas.

Exemple 2 Convertir $N = (0{,}732)_{10}$ en octal.

Solution Nous aurons successivement:

$0{,}732 \times 8 = \boxed{5}\ ,\ 856$

$0{,}856 \times 8 = \boxed{6}\ ,\ 848$

$0{,}848 \times 8 = \boxed{6}\ ,\ 784$

$0{,}784 \times 8 = \boxed{6}\ ,\ 272$

$0{,}272 \times 8 = \boxed{2}\ ,\ 176$

Donc $N = (0{,}732)_{10} = (0{,}56662)_8$

Vérification par l'expression polynomiale:

$$(0{,}56662)_8 \equiv 5 \times 8^{-1} + 6 \times 8^{-2} + 6 \times 8^{-3} + 6 \times 8^{-4} + 2 \times 8^{-5}$$

$$= 0{,}625 + 0{,}093750 + 0{,}011718750$$ en ne prenant que les trois premiers termes

$$= (0{,}730468750)_{10}$$

Ce qui est une bonne approximation.

1-4-4 CONVERSION BINAIRE-OCTAL ET VICE VERSA

Quatre chiffres binaires ou *bits* permettent $2^4 = 16$ combinaisons et donc d'écrire les seize entiers de 0 à 15 selon le tableau ci-contre.

0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Ce *code* est nommé code *binaire naturel* et dans notre cas plus spécifique code 8421. Chacun de ces chiffres représente le poids d'un bit. Ce code sera très souvent utilisé ultérieurement en techniques numériques: il serait donc utile qu'il vous soit familier dès à présent.

10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Nous allons maintenant illustrer comment convertir rapidement un nombre de base 2 en un nombre de base 8.

RÈGLE À partir de la virgule, grouper les bits par blocs de trois en allant vers la gauche pour la partie entière et vers la droite pour la partie fractionnaire. Convertir ensuite ces blocs en octal selon le code 8421.

Cette propriété découle du fait que la base 8 du système octal est une puissance entière de 2, en effet $8 = 2^3$.

Exemple 1

$$N = (\ \underline{\lfloor 110 \rfloor}\ \ \underline{\lfloor 111 \rfloor}\ \ \underline{\lfloor 011 \rfloor}\ ,\ \underline{\lfloor 001 \rfloor}\ \ \underline{\lfloor 101 \rfloor}\ )_2$$
$$= (\ \ 6\ \ \ \ 7\ \ \ \ 3\ \ ,\ \ 1\ \ \ \ 5\ \ )_8$$

Problème inverse:

Exemple 2 Convertir $N = (567{,}315)_8$ en binaire.

Solution Écrire, par blocs de trois bits, la valeur binaire des chiffres du nombre octal. On obtient dans ce cas:

$$N = (101\ \ 110\ \ 111\ ,\ 011\ \ 001\ \ 101)_2$$

Exemple 3 Convertir $N = (79182)_{10}$ en binaire.

Solution Convertissons ce nombre en nombre octal.

Division par 8	Quotient	Reste
79182 | 8		
71	9897	6
78		
62		
6		
9897 | 8		
18	1237	1
29		
57		
1		

Division par 16	Quotient	Reste
1237 |8		
43	154	5
37		
5		
154 |8		
74	19	2
2		
19 |8		
3	2	3
2 |8		
	0	2

Donc, $N = (79182)_{10}$

$= (232516)_8$

$= (010 \quad 011 \quad 010 \quad 101 \quad 001 \quad 110)_2$

C'est une méthode plus rapide que la conversion directe en binaire car le nombre de divisions par 2 est trois fois plus grand que celui des divisions par 8.

1-4-5 CONVERSION BINAIRE-HEXADÉCIMAL ET VICE VERSA

La base du système de numération hexadécimal est aussi une puissance entière de 2, en effet $16 = 2^4$.

On a donc les mêmes propriétés que pour le système octal, mais cette fois on groupe les bits par blocs de quatre.

Exemple 1

$N = (\underline{1001} \; \underline{1100} \; \underline{1011} \; \underline{1010} \; , \; \underline{0111})_2$

$= (\quad 9 \quad C \quad B \quad A \quad , \quad 7 \quad)_{16}$

Exemple 2 Convertir $N = (11432)_{10}$ en binaire.

Solution Convertissons ce nombre en nombre hexadécimal.

Division par 16	Quotient	Reste
11432 |16		
023	714	8
72		
08		

714	|16		
074		44	10
10			
44	|16		
12		2	12
2	|16		
		0	2

Donc $N = (11432)_{10}$

$= (2\ \ C\ \ A\ \ 8)_{16}$

$= (0010\ \ 1100\ \ 1010\ \ 1000)_2$

Exemple 3

$N = (A\ \ 7\ \ 8\ \ ,\ \ B\ \ 3\ \ 2)_{16}$

$= (1010\ \ 0111\ \ 1000\ \ ,\ \ 1011\ \ 0011\ \ 0010)_2$

1-5 OPÉRATIONS ARITHMÉTIQUES EN BINAIRE

1-5-1 L'ADDITION

REMARQUE: Multiplier un nombre décimal par $(10)_{10}$ revient à lui ajouter un 0. De la même façon, multiplier un nombre binaire par $(10)_2$ revient à lui ajouter un 0: $1 \times 10 = 10$; $101 \times 10 = 1010$.

Cela nous permet de faire une autre remarque: ajouter à un nombre son égal revient à le multiplier par $(10)_2$, donc à lui ajouter un 0: $1 + 1 = 10$; $101 + 101 = 1010$.

Cela provient du fait que:

$2^{n-1} + 2^{n-1} = 2\,(2^{n-1}) = 2^n$; ce qui se traduit, par exemple, en binaire par:

1000	+	1000	=	10000
(n − 1) zéros		(n − 1) zéros		n zéros

Dressons la table d'addition:

$0 + 0 = 0$

$0 + 1 = 1$

$1 + 0 = 1$

$1 + 1 = 0$ et report de 1

L'algorithme de l'addition des nombres binaires est le même que celui de l'addition des nombres décimaux.

Exemple 1

Reports:	1	1	1	1			
		1	1	0	1	0	1
+	1	0	1	1	0	0	
	1	1	0	0	0	0	1

Exemple 2

Reports:		1	1	1	1	1	1	1	1	1	1				
		1	0	1	1	1	0	0	1	1	1	1	1	0	0	1
+	1	1	0	1	1	1	1	0	1	0	1	0	1	1	0
	1	1	0	0	1	1	0	0	0	1	0	0	1	1	1	1

Pour 1 + 1 écrire 0 et reporter 1. Pour 1 + 1 et 1 de report, ce qui revient à 1 + 1 + 1, écrire 1 et reporter 1, ce qui revient à 11.

Il faut faire de nombreux exercices pour se familiariser avec l'addition en binaire.

1-5-2 LA SOUSTRACTION

Lorsque le diminuteur est plus petit que le diminuende, on aura un résultat de signe positif. Dans le cas contraire, intervertir les termes et affecter le résultat du signe − (moins).

Exemple en décimal:

$$\begin{array}{r} 632 \\ -\ 475 \\ \hline +\ 157 \end{array}$$

Au lieu d'effectuer la soustraction 475 − 632, effectuer 632 − 475 = 157 et écrire comme résultat − 157.

Dressons la table de soustraction:

0 − 0 = 0
0 − 1 = 1 et retenue de 1
1 − 0 = 1
1 − 1 = 0

La retenue de 1 sera retranchée du chiffre de rang supérieur.

Exemple 1

	0		0	0			
	~~1~~	0	~~1~~	~~1~~	0	1	1
−		1	0	1	1	1	1
		1	0	1	1	0	0

Lorsqu'on a une retenue, rayer le 1 de rang supérieur et le remplacer par 0.

Exemple 2

```
  0 1 1 0
  1 ~~0~~ ~~0~~ ~~1~~ 0 1 1
-   1 0 1 1 1 1
-------------------
    0 1 1 1 0 0
```

Dans ce cas, rayer pour la retenue le premier 1 rencontré, le remplacer par 0 et les 0 intermédiaires par des 1, car:

```
  1 0 0 0
-       1
---------
    1 1 1
```

1-5-3 LA MULTIPLICATION

La disposition des nombres à multiplier est la même en binaire qu'en décimal, l'algorithme est aussi le même.

La table de multiplication est particulièrement simple:

$0 \times 0 = 0$
$0 \times 1 = 0$
$1 \times 0 = 0$
$1 \times 1 = 1$

Si on multiplie par 1, écrire le multiplicande et si on multiplie par zéro, écrire 0.

Exemple 1

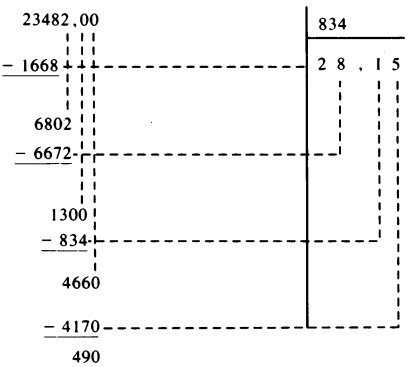
```
      1 0 1 1 0 1
  ×       1 0 1
-------------------
      1 0 1 1 0 1
1 0 1 1 0 1 .        <------- . décalage dû au zéro de 101
-------------------
1 1 1 0 0 0 0 1
```

Exemple 2

```
                    1 1 0 1 1 0 1
                  × 1 0 1 0 0 1 1
-------------------------------------
                    1 1 0 1 1 0 1
                  1 1 0 1 1 0 1
            1 1 0 1 1 0 1 . .
      1 1 0 1 1 0 1 .
-------------------------------------
1 0 0 0 1 1 0 1 0 1 0 1 1 1
```

1-5-4 LA DIVISION

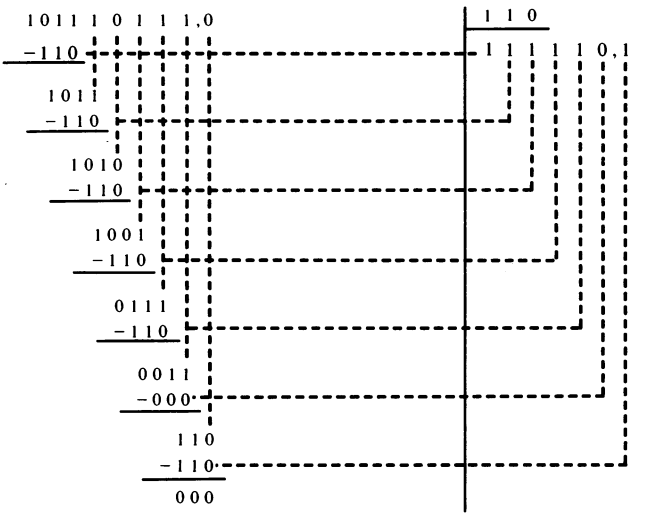
RAPPEL: soit à diviser $(23482)_{10}$ (dividende) par $(834)_{10}$ (diviseur). On pose la division:



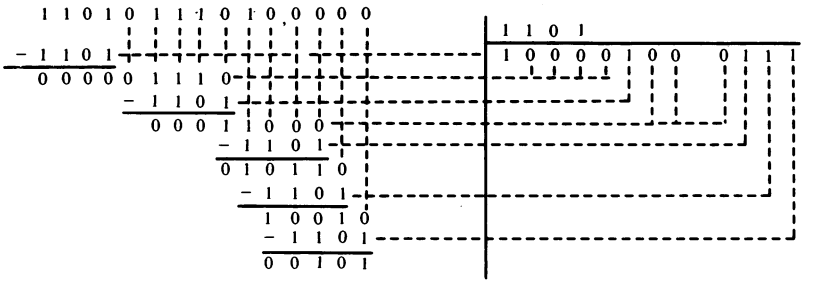
Cette disposition classique de la division est enseignée dans les classes élémentaires.

Pour la division en binaire, c'est la même disposition et le même algorithme. Les chiffres du quotient, plus simples, sont 0 ou 1; 1 lorsque le diviseur est plus petit que le dividende et 0 dans l'autre cas.

Exemple 1



Exemple 2



1-6 LA COMPLÉMENTATION

1-6-1 COMPLÉMENT À 1

En décimal, on forme le *complément à* 9 d'un nombre, par exemple 78543, en soustrayant de 9 chaque chiffre de ce nombre. Dans notre cas, on obtient 21456. La somme de ces nombres donne 99999.

En binaire, on forme le *complément à* 1 d'un nombre en soustrayant de 1 chaque bit de ce nombre. Le complément à 1 de 1 0 1 1 0 1 1 1 0 0 1 0, par exemple, est 0 1 0 0 1 0 0 0 1 1 0 1.

Donc, pour obtenir le complément à 1 d'un nombre binaire il suffit de complémenter chaque bit: lorsqu'on a 1, écrire 0 et lorsqu'on a 0, écrire 1. La somme d'un nombre binaire et de son complément à 1 est un nombre binaire uniquement composé de 1.

1-6-2 COMPLÉMENT À 2

En décimal, on forme le *complément à* 10 ou complément *vrai* d'un nombre, par exemple 673425, en soustrayant de 10 le chiffre de rang 0 et de 9 les autres. Dans notre cas on obtient 326575.

Effectuons la somme de ces deux nombres

$$\begin{array}{r} 673425 \\ +\ 326575 \\ \hline 1000000 \end{array}$$

on obtient:

Trouver le complément à 10 d'un nombre revient à le soustraire de la puissance de 10 immédiatement supérieure.

$$\begin{array}{r} 1000000 \\ -\ 673425 \\ \hline 326575 \end{array}$$

donne:

En binaire, trouver le *complément à* 2 d'un nombre revient à le soustraire de la puissance de 2 immédiatement supérieure. Par exemple le complément à 2 de

1 1 0 1 0 1 1 0 1 0 est le résultat de la soustraction

$$\begin{array}{r} 1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ -\ 1\,1\,0\,1\,0\,1\,1\,0\,1\,0 \\ \hline \end{array}$$

soit, 0 0 1 0 1 0 0 1 1 0

Trouver le complément à 2 revient aussi à trouver le complément à 1 (en complémentant chaque bit) et à ajouter 1 au résultat. Dans le cas précédent on obtiendrait:

$$\begin{array}{r} 0\,0\,1\,0\,1\,0\,0\,1\,0\,1 \\ +\qquad\qquad\qquad 1 \\ \hline 0\,0\,1\,0\,1\,0\,0\,1\,1\,0 \end{array}$$

Une troisième méthode consiste à conserver tous les bits à partir de la droite jusqu'au premier 1 compris et de changer les autres bits de 0 en 1 ou de 1 en 0 comme pour le complément à 1. Dans notre cas on aurait:

0 0 1 0 1 0 0 1 $\boxed{1\,0}$ bits conservés.

Exemple 1 Trouver le complément à 2 de 1 0 1 1 0 1 1 0 1 0 0 0

Réponse: 0 1 0 0 1 0 0 1 1 0 0 0

1-6-3 SOUSTRACTION PAR COMPLÉMENTATION À 1 ET ADDITION

Voyons maintenant une propriété du complément des nombres qui permettra de justifier un autre algorithme pour la soustraction. Nous tirerons cette propriété d'un exemple sur des nombres décimaux.

Soit la soustraction en décimal

$$\begin{array}{r} 17382 \\ -\ 12457 \\ \hline \end{array}$$

résultat: + 04925

Si nous prenons le complément à 9 du diminuteur, nous obtenons 87542 qui ajouté au diminuende donne:

$$\begin{array}{r} 1\,7\,3\,8\,2 \\ +\ 8\,7\,5\,4\,2 \\ \hline 1\,0\,4\,9\,2\,4 \\ \qquad\qquad +\ 1 \\ \hline \end{array}$$

et en ajoutant le dernier 1 on obtient

le résultat: 0 4 9 2 5 de la soustraction

Cette façon de procéder s'applique également aux nombres binaires en utilisant le complément à 1.

Exemple 1 Soit la soustraction suivante:

```
  1 1 0 1 0 1 1 1 0 1  --------→  1 1 0 1 0 1 1 1 0 1
- 1 0 1 1 1 0 0 1 1 1              + 0 1 0 0 0 1 1 0 0 0 : complément à 1
                                  1 0 0 0 1 1 1 0 1 0 1
                                  └──────────→ +     1
                                  ---------------------
                  résultat:       0 0 0 1 1 1 0 1 1 0
```

Exemple 2 Soit la soustraction suivante:

```
  1 0 1 0 0 1 1 1  --------→  1 0 1 0 0 1 1 1
- 1 0 0 1 1 0 0 1             + 0 1 1 0 0 1 1 0 : complément à 1
                              1 0 0 0 0 1 1 0 1
                              └──────────→ +   1
                              -----------------
                  résultat:   0 0 0 0 1 1 1 0
```

On peut déjà prévoir que l'on peut se passer d'ajouter 1 à la fin si on prend le complément à 2 du nombre à soustraire. Dans ce cas, on aura ajouté 1 au moment de la complémentation et le résultat sera le même.

1-6-4 SOUSTRACTION PAR COMPLÉMENTATION À 2 ET ADDITION

Exemple 1 Soit la soustraction suivante:

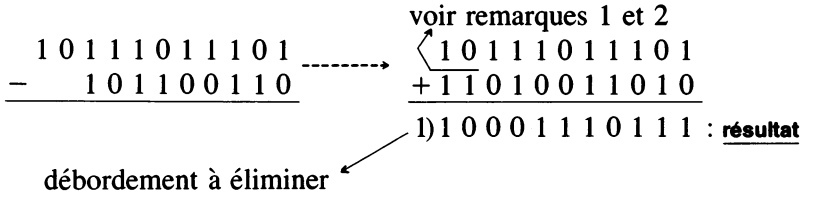
```
  1 1 0 1 1 0 1 1 1  --------→  1 1 0 1 1 0 1 1 1
- 1 0 1 0 1 1 1 0 1             + 0 1 0 1 0 0 0 1 1 : complément à 2
                                1)0 0 1 0 1 1 0 1 0 : résultat
                                ↙
débordement à éliminer
```

Exemple 2

REMARQUE 1: 11010 est équivalent à 0 0 0 1 1 0 1 0, les zéros en tête du nombre n'étant pas significatifs.

REMARQUE 2: Dans cet exemple on a remplacé les chiffres manquants en tête du nombre par des zéros, ce qui nous a donné des 1 au moment de la complémentation. Cette remarque est très importante sinon on ne retrouve pas le résultat de la soustraction.

Soit la soustraction suivante:



Nous allons étudier maintenant une forme d'*emmagasinage* de nombres dans un ordinateur.

1-6-5 FORME NORMALISÉE

Un nombre binaire placé dans une mémoire de machine peut être fractionnaire. Pour des raisons pratiques de circuiterie et de construction on est obligé de placer ces nombres sous *forme normalisée*. Il est clair qu'il est impossible de prévoir tous les cas possibles de position de la *virgule arithmétique*. Une des formes normalisées utilisées dans les machines automatiques est la représentation à *virgule fixe*.

Dans ce cas, pour des raisons d'efficacité, les machines sont construites pour représenter les nombres sous la forme:

+ 0 Λ .....

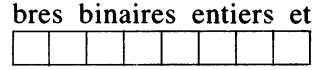
− 0 Λ ..... où Λ représente la virgule fixe différente parfois de la virgule arithmétique classique.

En décimal, par exemple, on peut normaliser les nombres de la façon suivante, à huit *caractères*

Nombres	Nombres normalisés à huit caractères
1282	+ Λ 1282000
31	+ Λ 0031000
7,4	+ Λ 0007400
0,032	+ Λ 0000032
− 1,32	− Λ 0001320

Le signe Λ n'est pas un caractère codé.

Le *facteur d'alignement* est le nombre de positions de caractères comprises entre la virgule fixe et la virgule arithmétique. Dans notre cas le facteur d'alignement est de 4.

Pour simplifier cet ouvrage nous ne considérerons que des nombres binaires entiers et nous choisirons un mot de huit caractères  dont une *case*, la première à gauche, sera réservée au signe. Le facteur d'alignement dans notre cas sera de 7.

Exemple 1 Représenter $(27)_{10}$ en binaire dans notre cas de normalisation.

Solution

$(27)_{10} = (11011)_2$ d'où:

+	0	0	1	1	0	1	1

Exemple 2 Même problème pour $(-37)_{10}$.

Solution

$(-37)_{10} = -(100101)_2$ d'où:

−	0	1	0	0	1	0	1

Exemple 3 Déterminer le plus grand nombre que l'on peut représenter de cette façon. Ce sera + 1 1 1 1 1 1 1 qui correspond à 64 + 32 + 16 + 8 + 4 + 2 + 1 = 127. Si nous voulons représenter un nombre plus grand que 127 nous aurons, dans notre cas, un débordement. Notre système ne comporte pas assez de cases pour placer un tel nombre.

Les gros ordinateurs ont jusqu'à 16 ou 32 bits pour les nombres entiers. Le miniordinateur PDP8 de Digital, par exemple, en a 12.

Les nombres fractionnaires ou les très grands nombres sont écrits sous forme d'une fraction normalisée et d'un exposant. On a alors une représentation à *virgule flottante*.

1-6-6 NOMBRES POSITIFS ET NÉGATIFS BINAIRES NORMALISÉS À HUIT CARACTÈRES

Les signes + et − ne sont pas assimilables tels quels par un ordinateur lequel ne connaît que deux états: 0 et 1. On convient donc de les représenter par un bit qui occupera la case de gauche du jeu de cases d'écriture du nombre considéré. Ce bit est appelé le *bit de signe*.

On peut, par exemple, représenter le signe + par 0 et le signe − par 1. C'est la convention généralement adoptée.

De plus, dans la plupart des cas, on représente les nombres négatifs sous la forme complément à 2.

C'est le mode d'écriture que nous retiendrons. Nous représenterons les nombres sous forme normalisée à huit caractères. Les nombres négatifs seront représentés sous leur forme complément à 2.

Exemple Représenter sous forme binaire normalisée à huit caractères les nombres décimaux.

Nombres	Solutions							
32	0	0	1	0	0	0	0	0
39	0	0	1	0	0	1	1	1
41	0	0	1	0	1	0	0	1
−27	1	1	1	0	0	1	0	1
−32	1	1	1	0	0	0	0	0
− 1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0

1-6-7 CALCUL AUTOMATIQUE

Grâce à cette façon de procéder, un calculateur additionne et soustrait automatiquement à l'aide d'un seul *circuit* dit *additionneur*. On obtiendra le résultat avec son signe. En voici quelques exemples.

Exemples Soit les additions suivantes, en représentations décimale à gauche, binaire normalisée à droite.

a)
$$\begin{array}{r} 27 \\ +\ 61 \\ \hline 88 \end{array} \qquad \begin{array}{r} 0\,0\,0\,1\,1\,0\,1\,1 \\ +\,0\,0\,1\,1\,1\,1\,0\,1 \\ \hline 0\,1\,0\,1\,1\,0\,0\,0 = (88)_{10} \end{array}$$

b)
$$\begin{array}{r} 61 \\ -\ 27 \\ \hline +\ 34 \end{array} \qquad \begin{array}{r} 0\,0\,1\,1\,1\,1\,0\,1 \\ +\,1\,1\,1\,0\,0\,1\,0\,1 \\ \hline 1)0\,0\,1\,0\,0\,0\,1\,0 = (34)_{10} \end{array}$$

débordement à éliminer ↗

c)
$$\begin{array}{r} 27 \\ -\ 61 \\ \hline -\ 34 \end{array} \qquad \begin{array}{r} 0\,0\,0\,1\,1\,0\,1\,1 \\ +\,1\,1\,0\,0\,0\,0\,1\,1 \\ \hline 1\,1\,0\,1\,1\,1\,1\,0 \end{array}$$

Le résultat est de signe −, en prenant le complément à 2 du résultat on a 000100010 donc − 34. On obtient donc le résultat cherché avec le signe − et sous la forme complément à 2.

d)
$$\begin{array}{r} 61 \\ +\ 88 \\ \hline 149 \end{array} \qquad \begin{array}{r} 0\,0\,1\,1\,1\,1\,0\,1 \\ +\,0\,1\,0\,1\,1\,0\,0\,0 \\ \hline 1\,0\,0\,1\,0\,1\,0\,1 = (149)_{10} \end{array}$$

Dans ce cas, on a un changement de signe du résultat. L'ordinateur détectera ce changement de signe et avertira l'opérateur qu'il y a débordement. C'est notre cas puisque 149 > 127, capacité maximale de nos *registres*.

Nous venons de voir comment un calculateur automatique effectue les quatre opérations élémentaires, la multiplication et la division se ramenant à des additions.

1-7 CODES

1-7-1 CODE GRAY OU BINAIRE RÉFLÉCHI

Le code binaire que nous avons vu jusqu'à maintenant s'appelle le code binaire naturel. Il existe de nombreux autres codes dont le code *Gray*, aussi appelé code *binaire réfléchi*. Dans les conversions d'une *grandeur analogique* (par exemple, la position de l'axe d'un moteur) en une *grandeur numérique* on a besoin d'un code dans lequel les grandeurs successives ne diffèrent que d'un caractère. Cela évite des *erreurs de détection*. Soit, le passage de 7 à 8, par exemple. Selon la règle binaire naturelle, on passera de 0111 à 1000: les quatre bits changent. Dans les états intermédiaires, s'ils ne changent pas tous en même temps, on peut détecter des valeurs erronées.

Le tableau ci-dessous donne l'équivalent en code Gray des entiers de 0 à 15.

0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
9	1101
10	1111
11	1110
12	1010
13	1011
14	1001
15	1000

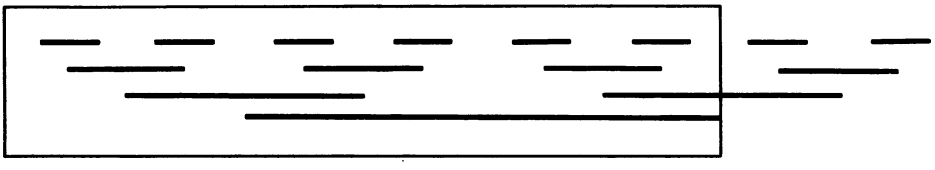
Ce code binaire est dit réfléchi, car $n-1$ de ses bits peuvent être générés par *réflexion* comme l'illustre le tableau suivant.

0 0	0 0 0	0 0 0 0
0 1	0 0 1	0 0 0 1
1 1	0 1 1	0 0 1 1
1 0	0 1 0	0 0 1 0
	1 1 0	0 1 1 0
	1 1 1	0 1 1 1
	1 0 1	0 1 0 1
	1 0 0	0 1 0 0
		1 1 0 0
		1 1 0 1
		1 1 1 1
		1 1 1 0
		1 0 1 0
		1 0 1 1
		1 0 0 1
		1 0 0 0

Ce tableau illustre une façon d'écrire un nombre décimal en Gray.

Une deuxième façon est donnée par la figure suivante qui est le schéma d'une règle codée en Gray. Les traits pleins peuvent correspondre à une fenêtre transparente et à des 1, l'absence de traits à une absence de fenêtres et à des 0. On détecte la lumière passant à travers ces fenêtres et on a immédiatement en code Gray la position du flux lumineux.

Le *codeur-décodeur* Gray de la figure suivante permettra, par exemple, de relever directement et sous forme numérique, la position longitudinale d'un chariot coulissant d'une machine-outil.



1-7-2 CODE BCD (Binary Coded Decimal) OU Décimal Codé Binaire (DCB)

Ce code conserve les avantages du système décimal et du code binaire. Il est utilisé par les machines à calculer.

On fait correspondre à chaque caractère du système décimal un mot du code binaire de quatre bits, on a alors:

code décimal	0	1	2	3	4	5	6	7	8	9
code BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

On remarque que si on supprime les zéros de gauche on retrouve le nombre binaire naturel.

Pour coder le nombre décimal 8 6 3 8 7, par exemple, écrire:

1000	0110	0011	1000	0111
8	6	3	8	7

Les opérations arithmétiques effectuées dans ce code sont plus compliquées qu'en binaire naturel. Par exemple:

9237	1001	0010	0011	0111
+ 81	+ 0000	+ 0000	+ 1000	+ 0001
9318	1001	0010	1011	1000
	9	2	?	8

Nous rencontrons ici un mot codé qui ne correspond pas à une valeur connue. Pour résoudre ce problème, on peut ajouter $(6)_{10} = (0\ 1\ 1\ 0)_2$ à ce mot codé inconnu, ce qui donnera

$$\begin{array}{r} 1011 \\ 0110 \\ \hline 10001 \end{array}$$

et ajouter 1 (report) au nombre suivant.

Dans ce cas, le résultat sera:

1001	0011	0001	1000
9	3	1	8

Cette difficulté provient du fait que quatre bits donnent $2^4 = 16$ états dont 10 seulement servent à coder les chiffres décimaux.

Dans les premiers ordinateurs on a cherché à contourner cette difficulté en utilisant le code «*plus trois*» étudié à la section 1-7-4.

1-7-3 CODE ASCII
(American Standard Code for Information Interchange)

Ce code est une norme presque universelle dans les *transmissions*. Il comprend sept ou huit caractères. Le huitième caractère dit de *parité* sert à détecter les erreurs de transmission. On étudiera ce mode de *détection d'erreurs* plus loin.

La table ci-dessous donne le code ASCII

b_7	0	0	0	0	1	1	1	1
b_6	0	0	1	1	0	0	1	1
b_5	0	1	0	1	0	1	0	1
b_4 b_3 b_2 b_1	0	1	2	3	4	5	6	7
0 0 0 0 0	NUL	TC_7 (DLE)	SP	0	@	P	`	p
0 0 0 1 1	TC_1 (SOH)	DC_1	!	1	A	Q	a	q
0 0 1 0 2	TC_2 (STX)	DC_2	"	2	B	R	b	r
0 0 1 1 3	TC_3 (ETX)	DC_3	#	3	C	S	c	s
0 1 0 0 4	TC_4 (EOT)	DC_4	¤	4	D	T	d	t
0 1 0 1 5	TC_5 (ENQ)	TC_8 (NAK)	%	5	E	U	e	u
0 1 1 0 6	TC_6 (ACK)	TC_9 (SYN)	&	6	F	V	f	v
0 1 1 1 7	BEL	TC_{10} (ETB)	'	7	G	W	g	w
1 0 0 0 8	FE_0 (BS)	CAN	(	8	H	X	h	x
1 0 0 1 9	FE_1 (HT)	EM	)	9	I	Y	i	y
1 0 1 0 10	FE_2 (LF)	SUB	*	:	J	Z	j	z
1 0 1 1 11	FE_3 (VT)	ESC	+	;	K	[	k	{
1 1 0 0 12	FE_4 (FF)	IS_4 (FS)	,	<	L	\	l	|
1 1 0 1 13	FE_5 (CR)	IS_3 (GS)	-	=	M	]	m	}
1 1 1 0 14	SO	IS_2 (RS)	.	>	N	^	n	~
1 1 1 1 15	SI	IS_1 (US)	/	?	O	_	o	DEL

CCITT-11540

Exemples de codes Le code binaire est donné par $b_7\ b_6\ b_5\ b_4\ b_3\ b_2\ b_1$, ($b_1$ est le bit de poids faible).

Le code de:

U est 55_{16} soit 1010101 en binaire
F est 46_{16} soit 1000110
K est $4B_{16}$ soit 1001011

réciproquement, le code correspondant à:

43_{16} ou 1000011 est C
0110011 soit 33_{16} est 3

1-7-4 CODE «PLUS TROIS» *(Excess 3)*

L'inconvénient cité ci-dessus à propos des opérations arithmétiques en BCD a été contourné dans les ordinateurs de la première génération à l'aide du code «plus trois». Le code plus trois des nombres décimaux de 0 à 9 est le code BCD auquel on ajoute 3, comme son nom l'indique, ce qui donne:

0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100

En général, pour avoir le résultat d'une addition en BCD + 3 il faut ajouter 3 si on a un report et retrancher 3 dans le cas contraire (−3 = −0011 = 1101 en complément à 2).

Exemple 1 Avec report

	devint en BCD plus trois			
7			1010	
+ 5		+	1000	
12		0001	0010	
		0011	0011	
		0100	0101	: BCD + 3

Exemple 2 Sans report

```
    5  devient en BCD plus trois  1000
  + 3                             0110
  ---                            ------
    8                             1110
                                +  1101
                                ------
                                1) 1011   : BCD + 3
```

débordement à éliminer ——————↗

1-7-5 CODES DÉTECTEURS D'ERREURS

Dans un code de quatre bits par exemple (ou de sept bits pour le code ASCII) une erreur simple sur un bit peut faire apparaître un autre *mot de code*. Par exemple, en BCD le mot de code 0010 (2) transmis par erreur 0110 (6, inversion du 3^{e} bit due aux parasites de la ligne) sera interprété comme un 6 par le *récepteur*, mais ce sera une *donnée* erronée.

Il existe plusieurs façons de détecter ce type d'erreurs. La plus utilisée est celle du *bit de parité paire* ou *impaire*.

Exemple de code de parité paire en BCD:

	8421	p
0	0000	0
1	0001	1
2	0010	1
3	0011	0
4	0100	1
5	0101	0
6	0110	0
7	0111	1
8	1000	1
9	1001	0

p: bit de parité paire
p = 0 si le nombre de 1 du mot de code est pair
= 1 autrement.

Si on a une erreur simple dans le mot de code ou dans le bit de parité, elle sera immédiatement détectée. Le récepteur pourra demander une retransmission jusqu'à réception du bon mot.

Il existe d'autres codes de détection d'erreurs. Citons, par exemple, le code «*deux de cinq*» qui représente les dix combinaisons possibles de deux 1 dans un mot de cinq bits, ce qui donne:

0	00011
1	11000
2	10100

3	01100
4	10010
5	01010
6	00110
7	10001
8	01001
9	00101

1-7-6 CODES DÉTECTEURS ET CORRECTEURS D'ERREURS

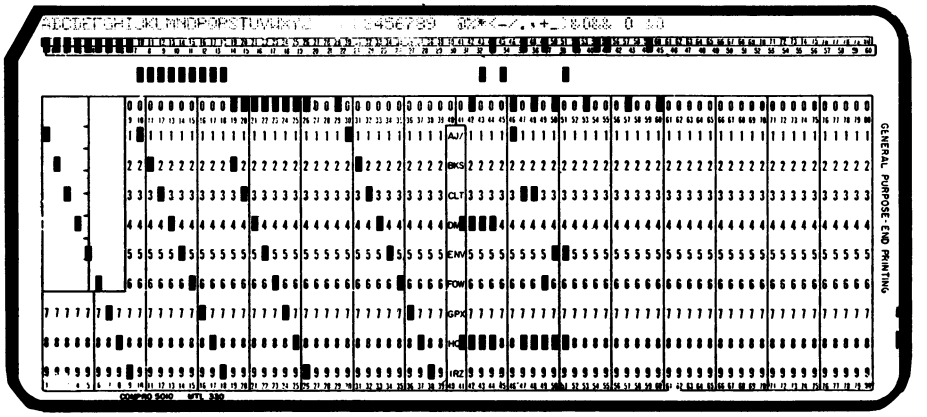
Le système *correcteur d'erreurs* suivant est utilisé sur les *bandes magnétiques*. On envoie une série de mots codés (verticalement sur le tableau) et en fin de *message* on envoie les parités paires *longitudinale* et *verticale*.

	3	4	5	7	parité longitudinale
	0	0	0	0	0
	0	1	1	1	1
	1	0	0	1	0
	1	0	1	1	1
parité verticale	0	1	0	1	

Si, par exemple, le mot de code de 5 comporte une erreur, disons que son 2e bit est erroné, la machine localisera une erreur dans la 3e colonne et dans la 2e ligne et la corrigera immédiatement.

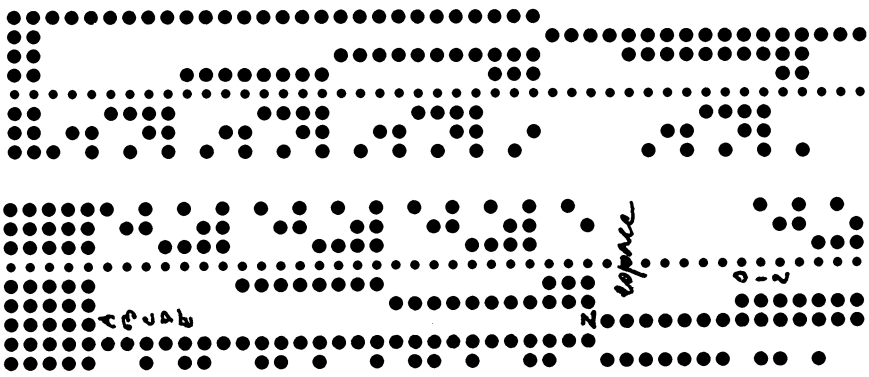
1-8 CODAGE

1-8-1 CARTE PERFORÉE SELON LE CODE HOLLERITH



Carte perforée selon le code *Hollerith*

1-8-2 RUBAN PERFORÉ SELON LE CODE ASCII



Disposition des perforations. Un trou représente un 1.

o o o o o . o o o

↑ perforation pour l'entraînement

↑ bit de parité

Exemple Le code de la lettre «B» qui est 42_{16} sera représenté par

. o . . . . . o .

1-8-3 IMPULSIONS

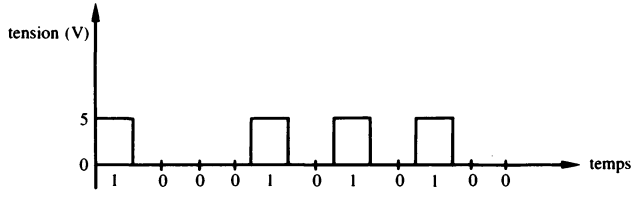
Pour représenter l'*état* 0 ou 1 d'une donnée à transmettre, les calculateurs utilisent deux *niveaux de tension*. Les plus répandus sont

0 volt pour l'état 0
5 volts pour l'état 1.

Cette *logique* est connue sous le nom de logique *positive* (l'inverse, 0 volt pour l'état 0 et −5 volts pour l'état 1 est appelé logique *négative*).

a) Impulsions série

Pour transmettre un nombre, on envoie donc dans le temps deux niveaux ou *impulsions* de tension selon, par exemple, la configuration *série* suivante:

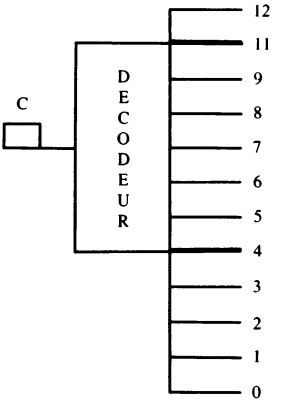


Ces tensions sont soit positives soit négatives. L'important est d'avoir deux niveaux de tension nettement distincts.

On n'a donc besoin dans ce cas que d'un fil de transmission d'où une économie de câblage. Cela présente un inconvénient: la transmission d'un mot s'étire dans le temps.

b) Impulsions parallèle

Lorsqu'on veut actionner les marteaux d'une perforatrice, par exemple, on actionne tous les marteaux nécessaires d'une même colonne en même temps. Pour cela on commande les électro-aimants par des impulsions *parallèle*. La figure ci-dessous illustre la perforation de C en parallèle.



Avantage d'une transmission par impulsions parallèle:
— Rapidité: on a tout le mot codé en même temps.

Inconvénient:
— Nécessite beaucoup de fils de transmission. En général plus coûteuse.

PROBLÈMES

Voir le corrigé des problèmes à l'appendice A. Si votre réponse diffère de celle y figurant, rechercher l'erreur commise.

1-1 Écrire les nombreux décimaux suivants sous forme polynomiale:

a) 27; b) 301;

c) 08641; d) 39472;

e) 473283.

1-2 Déterminer le chiffre de poids fort des nombres suivants:

a) 3812; b) 0073432;

c) 162622.

1-3 Déterminer le chiffre de poids faible des nombres suivants:

a) 7432; b) 3890;

c) 170003.

1-4 Soit le nombre 49736421, déterminer le rang de:

a) 9; b) 6;

c) 2; d) 1;

e) 7; f) 4.

1-5 Donner, sous forme d'un ensemble ordonné, les symboles que vous utiliseriez dans les systèmes de numération de base:

a) 3; b) 7;

c) 4; d) 11;

e) 5; f) 13;

g) 16.

1-6 Décomposer les nombres suivants en fonction des puissances de la base de leur système de numération:

a) $(101101110)_2$; b) $(B9107A)_{12}$;

c) $(EA735)_{16}$; d) $(654321)_7$;

e) $(6583)_8$.

1-7 Calculer la valeur décimale des nombres suivants:

a) $(CD1)_{16}$; b) $(101110101)_2$;

c) $(6731)_8$; d) $(1010)_{16}$;

e) $(41)_8$.

1-8 Calculer la valeur décimale des nombres suivants:

a) $(553)_6$; b) $(327)_8$;

c) $(1AB)_{12}$; d) $(CD)_{16}$.

1-9 Selon le premier procédé de conversion, convertir en octal et en hexadécimal les nombres décimaux suivants:

a) 47934; b) 91132;

c) 743; d) 121.

1-10 Selon le premier procédé de conversion, convertir en binaire les nombres décimaux suivants:

a) 283; b) 151;

c) 1142.

1-11 Selon le deuxième procédé de conversion, convertir en octal et en hexadécimal les nombres décimaux suivants;

a) 31432; b) 87947;

c) 653; d) 831.

1-12 Selon le deuxième procédé de conversion, convertir en binaire les nombres décimaux suivants:

a) 253; b) 432;

c) 512; d) 1024.

1-13 Déterminer l'équivalent décimal des nombres binaires suivants:

a) 11,011011; b) 1,101101;

c) 1,0001011.

1-14 Déterminer l'équivalent décimal des nombres de base 8 suivants:

a) 0,252; b) 0,347;

c) 0,151.

1-15 Convertir en binaire, en octal et en hexadécimal les nombres décimaux suivants:

a) 0,375; b) 0,432;

c) 0,741.

1-16 Convertir les nombres décimaux suivants en octal puis directement en binaire:

a) 123,431; b) 241,172.

1-17 Convertir les nombres décimaux suivants en hexadécimal puis directement en binaire:

a) 653; b) 831;

c) 87947.

1-18 Convertir en hexadécimal et en octal les nombres binaires suivants:

a) 110111,0110111101; b) 1110001111,010101;

c) 101110111011,0101111.

1-19 Effectuer les additions suivantes en binaire:

a)
$$\begin{array}{r} 1011{,}1101 \\ +\ 110{,}1011 \\ \hline \end{array}$$

b)
$$\begin{array}{r} 110111{,}01 \\ +\ 110011{,}11 \\ \hline \end{array}$$

c)
$$\begin{array}{r} 101111101 \\ +\ 111111011 \\ \hline \end{array}$$

d)
$$\begin{array}{r} 1101111 \\ +\ 1110000 \\ \hline \end{array}$$

1-20 Effectuer les soustractions suivantes en binaire:

a)
$$\begin{array}{r} 1011{,}1101 \\ -\ 110{,}1011 \\ \hline \end{array}$$

b)
$$\begin{array}{r} 110111{,}01 \\ -\ 110011{,}11 \\ \hline \end{array}$$

c)
$$\begin{array}{r} 111111011 \\ -\ 101111101 \\ \hline \end{array}$$

d)
$$\begin{array}{r} 1110000 \\ -\ 1101111 \\ \hline \end{array}$$

1-21 Effectuer les multiplications suivantes en binaire:

a)
$$\begin{array}{r} 11011{,}01 \\ \times\ 101{,}01 \\ \hline \end{array}$$

b)
$$\begin{array}{r} 10111{,}11 \\ \times\ 110 \\ \hline \end{array}$$

c)
$$\begin{array}{r} 11001{,}011 \\ \times\ 1110 \\ \hline \end{array}$$

d)
$$\begin{array}{r} 110111 \\ \times\ 10001 \\ \hline \end{array}$$

1-22 Effectuer les divisions suivantes en binaire:

a) 101101 | 1101 b) 1101111 | 1010

c) 11011101 | 1001 d) 11011 | 11

1-23 Trouver: A) le complément à 1; B) le complément à 2 des nombres binaires suivants:

a) 1101110111; b) 1011011101; c) 11011;

d) 10110111; c) 1000001; f) 1100111;

g) 110111; h) 10110101; i) 01001110.

1-24 Effectuer les soustractions suivantes en utilisant le complément à 1:

a) 1011,1101 b) 110111,01 c) 111111011

− 110,1011 − 110011,11 − 101111101

1-25 Effectuer les soustractions suivantes en utilisant le complément à 2:

a) 1110000 b) 1101111 c) 100111101

− 1101111 − 1011101 − 11011110

1-26 Écrire sous forme normalisée binaire, à huit caractères, en complémentant à 2 les nombres négatifs, les nombres décimaux suivants:

a) −17; b) 31;

c) −42; d) 51;

e) −128.

1-27 Inclure le bit de signe des nombres ci-dessus.

1-28 Donner sous forme normalisée le résultat des opérations suivantes en indiquant son équivalent décimal.

a) (31 − 17); b) (42 − 51);

c) (− 128 − 17); d) (31 − 42).

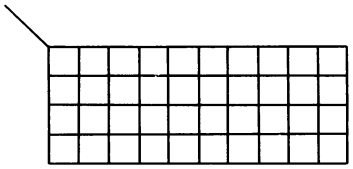
1-29 Écrire en code binaire réfléchi (Gray) les nombres suivants:

a) 27; b) 31;

c) 19; d) 23.

1-30 Coder en binaire réfléchi

a) les colonnes; b) les lignes de la table suivante:



1-31 Écrire en B C D les nombres décimaux suivants:

a) 8732; b) 4149;

c) 7032; d) 4096.

1-32 Écrire l'équivalent décimal des nombres binaires B C D suivants:

a) 0111 0101 1001 1000 0110;

b) 0010 0101 0111 0001 0001.

1-33 Donner le code ASCII de F.

1-34 Donner le symbole correspondant à 1001110 en ASCII.

1-35 Donner le nombre standard de colonnes d'une carte perforée.

1-36 Faire le schéma de la lettre A sur un ruban perforé en code ASCII (huit *moments*).

1-37 Déterminer le nombre de fils requis pour envoyer le nombre 11010: a) en série; b) en parallèle et c) dire quel mode de transmission est le plus rapide.

2

Algèbre de Boole

2-1 OBJECTIFS

1. Savoir définir l'ensemble d'application de l'algèbre de Boole ainsi que les trois opérations de base: Négation, ET (intersection), OU (union). Connaître la table de vérité de chacune de ces opérations et leur symbole graphique (porte).
2. Connaître les huit lois fondamentales de l'algèbre de Boole, savoir implanter chacune d'elles à l'aide de portes, savoir les prouver par une table de vérité et savoir les appliquer.
3. Savoir dresser la table de vérité d'une fonction logique et savoir l'implanter.
4. Savoir donner une définition sous forme algébrique ou d'une table de vérité des opérations NON-OU, OU exclusif et NON-ET.
5. Savoir démontrer, implanter et appliquer les relations de base de l'algèbre de Boole.
6. Connaître les deux théorèmes de De Morgan, savoir les prouver, les appliquer et les implanter.
7. Savoir utiliser la dualité de l'algèbre de Boole pour transposer une relation en une autre.
8. Savoir simplifier algébriquement une fonction logique.

2-2 INTRODUCTION

Nous allons étudier maintenant une algèbre semblable, sous certains aspects, à l'algèbre «classique» dont elle diffère cependant de manière caractéristique. D'importantes applications du domaine des ordinateurs et des appareils de mesure numériques reposent sur elle. Cette algèbre, objet de ce chapitre, porte divers noms: algèbre des propositions, algèbre de la logique ou, le plus souvent, *algèbre de Boole* son inventeur vers 1850.

L'algèbre de Boole est un ensemble de variables à deux états, de *valeurs de vérité* 1 (*vrai*), 0 (*faux*), muni d'un nombre limité d'*opérateurs*: NON, ET, OU. La manipulation de ces *variables* dites *booléennes* à l'aide de ces opérateurs donne des *fonctions*, booléennes elles-aussi, car leur résultat est une variable booléenne.

Nous verrons ci-dessous les opérateurs et les lois fondamentales, ou axiomes, sur lesquelles reposent cette algèbre. La variable «le circuit est

ouvert» vraie (1) ou fausse (0), selon le cas, est un exemple de variable booléenne.

2-3 OPÉRATIONS, OU FONCTIONS, DE BASE DE L'ALGÈBRE DE BOOLE

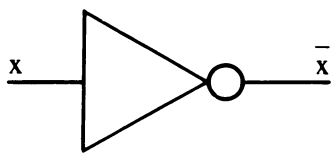
2-3-1 OPÉRATION À UNE VARIABLE: OPÉRATION NON

Soit x une variable booléenne; sa *négation*, NON x, appelée aussi *complément* de x, sera notée $\bar{x}$ (lire x barre). NON x sera également une variable booléenne.

Il est commode d'indiquer sous forme de tables les valeurs des variables soumises aux opérateurs et ce pour toutes les *combinaisons* possibles des valeurs de ces variables. Ces tables s'appellent des *tables de vérité*. Nous les introduirons de façon axiomatique. Par définition x peut prendre la valeur 0 ou 1. On aura pour l'opération NON la table de vérité ci-dessous. L'on voit que $\bar{x}$ est une variable booléenne «inversée» par rapport à x, d'où le nom d'*inverseur* donné au dispositif effectuant cette opération appelée aussi de ce fait *inversion*.

Entrée	Sortie
x	$\bar{x}$
0	1
1	0

Table de vérité



Symbole graphique

Pour concrétiser le cours, nous choisirons une technologie, parmi beaucoup d'autres, d'implantation des fonctions de base étudiées. La technologie retenue est le *circuit intégré* TTL série 74 de la société Texas Instruments Incorporated. C'est la plus répandue et, de plus, elle se prête bien à l'expérimentation. TTL est l'abréviation de «*Transistor Transistor Logic*». Toutes les fonctions sont *implantées* à l'aide de «*portes*» de la série TTL logées dans ses boîtiers rectangulaires normalisés comportant, pour la plupart, quatorze *broches*. La tension d'alimentation est de +5 V par rapport à la masse. Les tensions de sortie possèdent deux niveaux: un *niveau bas* «L» compris entre 0 et 0,8 volt et un *niveau haut* «H» compris entre 3 et 5 volts.

Le circuit 7404 d'*implantation* de l'inversion ou négation comporte six inverseurs. L'assignation des broches est donnée par le *schéma de brochage* de la figure 2-1. (Tous les schémas proviennent du catalogue de la société Texas Instruments Incorporated).

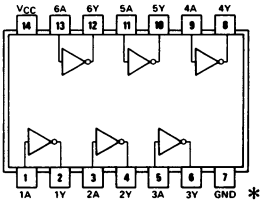


Figure 2-1 Implantation de l'inversion par le CI 7404 comprenant six inverseurs.

2-3-2 OPÉRATIONS À DEUX VARIABLES

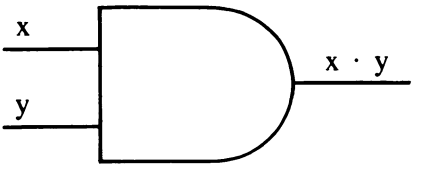
2-3-2-1 Opération ET

Soit x et y deux variables booléennes. Le résultat de x ET y est, selon la table de vérité ci-dessous, une variable booléenne.

Entrées		Sortie
x	y	x ET y
0	0	0
0	1	0
1	0	0
1	1	1

Les combinaisons possibles des entrées apparaissent dans l'ordre binaire naturel.

L'opérateur ET est aussi noté « · », la table justifie cette notation, d'où x ET y = x · y. Le résultat peut être désigné par une autre lettre, z par exemple, d'où l'on aura x ET y = z.



Symbole graphique

*Dans le but de préserver, dans toute la mesure du possible, l'intégrité des schémas et fiches signalétiques de la société Texas Instruments Incorporated, nous laisserons tels quels les termes anglais y figurant. Nous prions le lecteur peu familiarisé avec ces expressions de se reporter au lexique anglais-français figurant à l'appendice B.

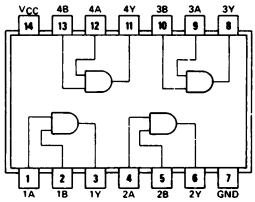


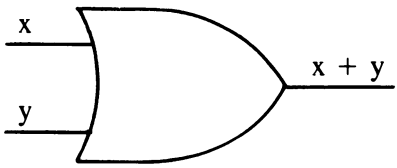
Figure 2-2 Implantation de la fonction ET par le CI 7408 comprenant quatre portes ET à deux *entrées*.

2-3-2-2 Opération OU

Soit x et y deux variables booléennes. Le résultat de x OU y est, selon la table de vérité ci-dessous, une variable booléenne.

x	y	x OU y
0	0	0
0	1	1
1	0	1
1	1	1

L'opérateur OU est aussi noté « + », on aura donc x OU y = x + y. Le résultat peut être désigné par une autre lettre, z, par exemple, d'où l'on aura x OU y = z.



Symbole graphique

REMARQUE Éviter de confondre l'opérateur + et l'opérateur ET. En langage courant, l'expression «2 et 3 font 5» signifie «2 + 3 = 5» tandis qu'en logique «x ET y» est une *intersection* revenant à une *multiplication*. Le résultat de l'opération ET est 1 si et seulement si tous les opérandes valent 1. Le résultat de l'opération OU est 1 si au moins l'un des opérandes vaut 1. L'opération OU est une *union* revenant à une *addition*.

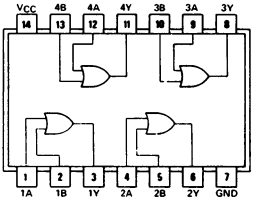
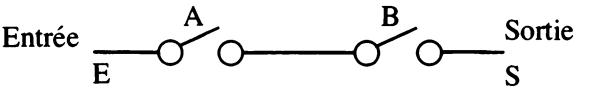


Figure 2-3 Implantation de la fonction OU par le CI 7432 comprenant quatre portes OU à deux entrées.

2-4 APPLICATION À UN RÉSEAU ÉLECTRIQUE

L'algèbre de Boole permet d'analyser ou de synthétiser un réseau de contacts électriques. Chaque contact est désigné par une lettre A, B, C, . . . Si deux contacts sont jumelés, ils porteront la même lettre. Si les deux sont ouverts ou fermés en même temps, on les désignera simplement par la même lettre. Par contre, si l'un est ouvert pendant que l'autre est fermé on les désignera par la même lettre mais l'une sera le complément de l'autre. Par exemple:

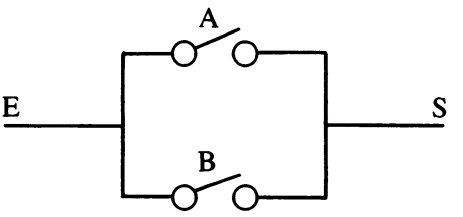
Les contacts peuvent être montés en série:



Si on a une tension en E on n'aura une tension en S que si A et B sont fermés. On notera ce cas par l'équation:

$$S = A \cdot B : \text{ET logique}$$

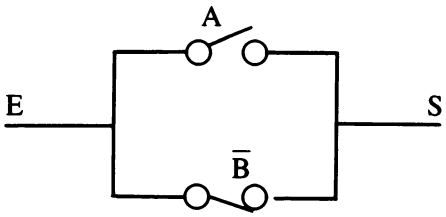
Si les contacts sont montés en parallèle on aura le schéma:



Si on a une tension en E, on aura une tension en S si A ou B est fermé ou si les deux le sont. Ce cas sera représenté par l'équation:

$$S = A + B : \text{OU logique}$$

Si un des contacts est normalement fermé, il sera désigné par sa lettre complémentée. Le réseau ci-dessous, par exemple, aura pour équation $S = A + \bar{B}$.



2-5 AXIOMES OU LOIS FONDAMENTALES DE L'ALGÈBRE DE BOOLE

Soit A, B et C trois variables booléennes, les opérateurs NON, ET et OU et les tables de vérité axiomatiques vues ci-dessus. Nous pouvons dès lors énoncer les lois suivantes vérifiables par les tables de vérité correspondantes.

2-5-1 LOIS DE FERMETURE

a) $A \cdot B$ est une variable booléenne définie par la table de vérité de l'opération ET vue ci-dessus.

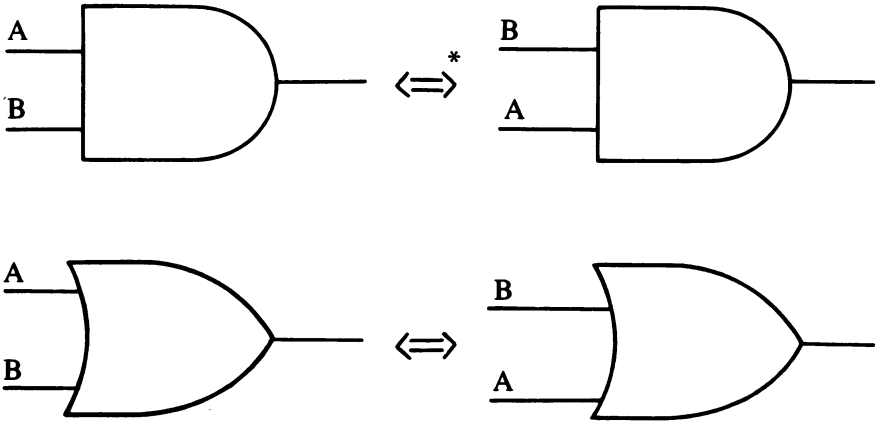
b) $A + B$ est une variable booléenne définie par la table de vérité de l'opération OU vue ci-dessus.

2-5-2 LOIS DE COMMUTATIVITÉ

a) $A \cdot B = B \cdot A$

b) $A + B = B + A$

L'observation des tables de vérité nous donne ce résultat immédiat. On n'aura donc pas à distinguer les entrées des portes. Symboliquement on aura:



2-5-3 LOIS D'ASSOCIATIVITÉ

a) $A \cdot (B \cdot C) = (A \cdot B) \cdot C$

b) $A + (B + C) = (A + B) + C$

Vérifions la loi b) par *induction parfaite*, c'est-à-dire par vérification dans tous les cas possibles en utilisant les tables de vérité:

A	B	C	B + C	A + (B + C)	A + B	(A + B) + C
0	0	0	0	0	0	0
0	0	1	1	1	0	1
0	1	0	1	1	1	1
0	1	1	1	1	1	1
1	0	0	0	1	1	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

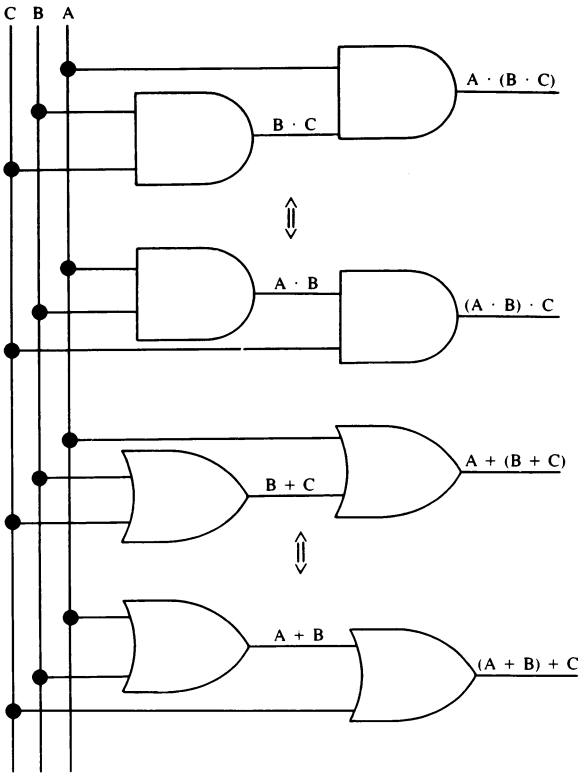
Entrées en binaire naturel

Identité de ces deux colonnes. On a donc vérifié que $A + (B + C) = (A + B) + C$

*Mis pour équivalent à

On peut vérifier de la même façon que: $A \cdot (B \cdot C) = (A \cdot B) \cdot C$

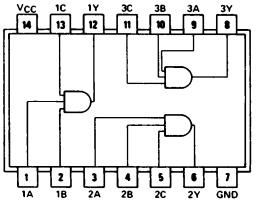
Cela nous permet de faire les schémas des circuits à trois entrées ci-dessous.



On peut donc écrire respectivement les fonctions · et + à trois variables sous les formes: P = ABC et S = A + B + C puisque le regroupement des variables n'a pas d'importance.

Dans la série TTL, le CI 7411 comporte trois portes ET à trois entrées et le CI 7421 en comporte deux à quatre entrées.

7411



7421

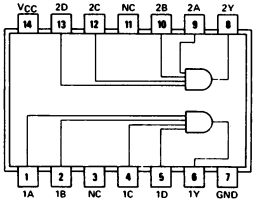


Figure 2-4 Portes ET à trois entrées (7411) et à quatre entrées (7421)

2-5-4 LOIS DE DISTRIBUTIVITÉ

a) de l'opération ET sur l'opération OU

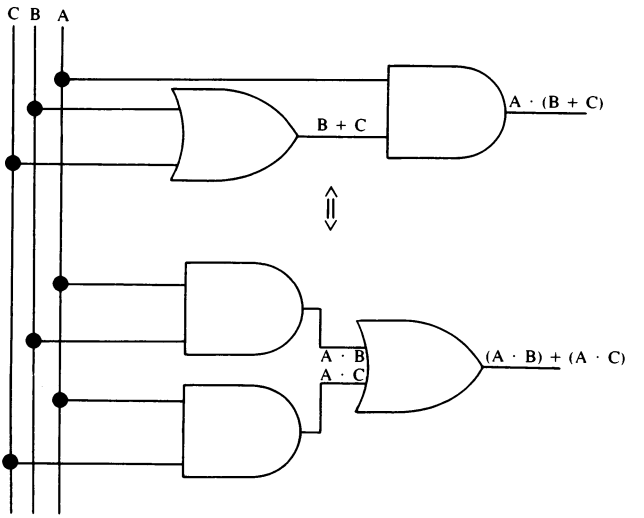
$$A \cdot (B + C) = (A \cdot B) + (A \cdot C)$$

b) de l'opération OU sur l'opération ET

$$A + (B \cdot C) = (A + B) \cdot (A + C)$$

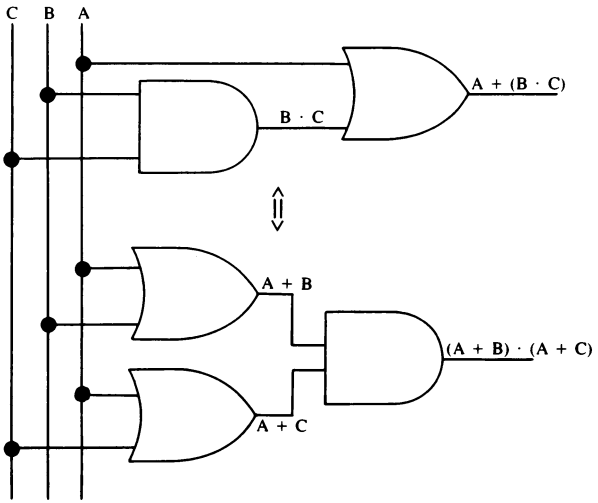
La loi d'écriture de ces relations dans l'autre sens s'appelle la *mise en facteur* ou *en évidence*.

EXERCICE Vérifier ces deux lois par induction parfaite (tables de vérité). Les deux montages ci-dessous sont *fonctionnellement identiques* (1re loi de distributivité).



Il est important de noter que le deuxième circuit exige trois portes au lieu de deux pour le premier. Il n'est donc pas indifférent de réaliser les circuits d'une façon ou d'une autre. Pour minimiser les coûts, il peut être utile de minimiser le nombre de portes.

La même remarque s'applique aux circuits suivants fonctionnellement identiques (2e loi de distributivité).



Par induction parfaite, puisque la variable booléenne A ne peut prendre que les valeurs 0 ou 1, on a les lois 5, 6 et 7 suivantes:

2-5-5 LOIS D'IDEMPOTENCE

a) $A + A = A$, en effet:

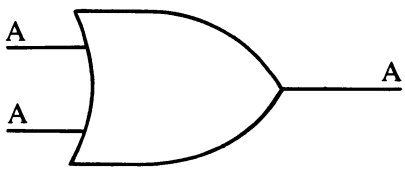
$$\begin{array}{l} A + A \\ 0 + 0 = \boxed{0} \\ 1 + 1 = \boxed{1} \end{array} = A$$

b) $A \cdot A = A$, en effet:

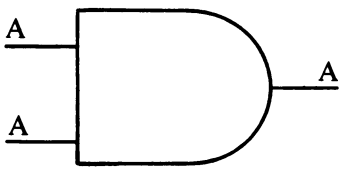
$$\begin{array}{l} A \cdot A \\ 0 \cdot 0 = \boxed{0} \\ 1 \cdot 1 = \boxed{1} \end{array} = A$$

Symboliquement:

a)



b)



2-5-6 LOIS DE COMPLÉMENTARITÉ

a) $A + \bar{A} = 1$, en effet:

$$A + \bar{A}$$
$$\begin{matrix} 1 + 0 = \\ 0 + 1 = \end{matrix} \boxed{\begin{matrix} 1 \\ 1 \end{matrix}} = 1$$

b) $A \cdot \bar{A} = 0$, en effet:

$$A \cdot \bar{A}$$
$$\begin{matrix} 0 \cdot 1 = \\ 1 \cdot 0 = \end{matrix} \boxed{\begin{matrix} 0 \\ 0 \end{matrix}} = 0$$

2-5-7 IDENTITÉS REMARQUABLES

a) $1 \cdot A = A$, en effet:

$$1 \cdot \bar{A}$$
$$\begin{matrix} 1 \cdot 0 = \\ 1 \cdot 1 = \end{matrix} \boxed{\begin{matrix} 0 \\ 1 \end{matrix}} = A$$

b) $1 + A = 1$, en effet:

$$1 + A$$
$$\begin{matrix} 1 + 0 = \\ 1 + 1 = \end{matrix} \boxed{\begin{matrix} 1 \\ 1 \end{matrix}} = 1$$

c) $0 \cdot A = 0$, en effet:

$$0 \cdot A$$
$$\begin{matrix} 0 \cdot 0 = \\ 0 \cdot 1 = \end{matrix} \boxed{\begin{matrix} 0 \\ 0 \end{matrix}} = 0$$

d) $0 + A = A$, en effet:

$$0 + A$$
$$\begin{matrix} 0 + 0 = \\ 0 + 1 = \end{matrix} \boxed{\begin{matrix} 0 \\ 1 \end{matrix}} = A$$

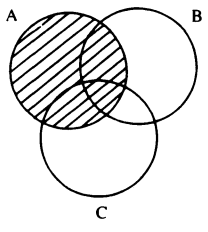
REMARQUE. Les circuits intégrés TTL présentent la caractéristique suivante: laisser une entrée d'une porte ouverte, (c'est-à-dire en l'air: non reliée à une tension ou à la masse) revient à avoir un 1 à cette entrée. Cette remarque est très utile pour les séances de laboratoire.

2-5-8 LOIS DE DISTRIBUTIVITÉ INTERNE

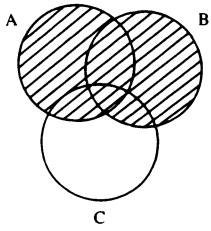
a) $A + (B + C) = (A + B) + (A + C)$

b) $A \cdot (B \cdot C) = (A \cdot B) \cdot (A \cdot C)$

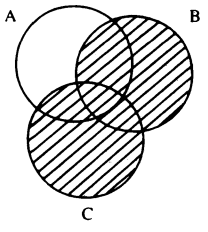
Autre façon de vérifier ces lois: par la méthode des *diagrammes de Venn*. On aura, par exemple:



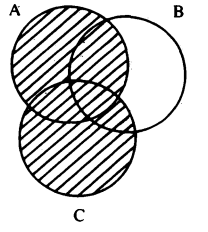
A



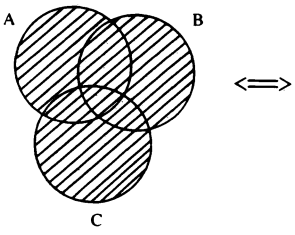
A + B



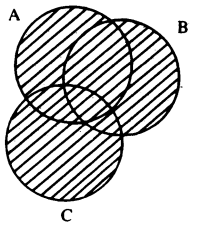
B + C



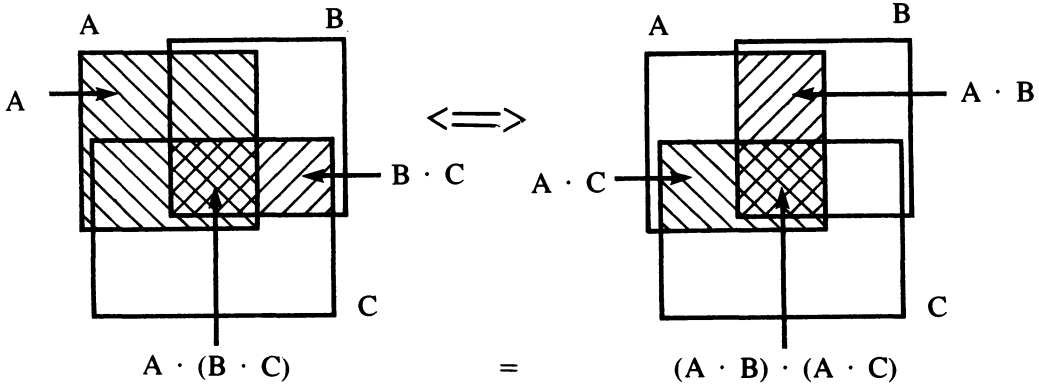
A + C



A + (B + C) =



(A + B) + (A + C)



2-6 ÉVALUATION D'UNE FONCTION LOGIQUE

Définition On appelle *fonction logique* une combinaison de variables booléennes reliées par les opérateurs «NON, ET, OU».

Exemples $z = (x + \bar{y})\,(x + y)\;)\;t(\bar{x} + y)$

$S = (A + B)\,(A + \bar{C}) + C(\bar{A} + B)$

Ces fonctions logiques peuvent se simplifier à l'aide des lois énoncées à la section précédente et s'évaluer à l'aide des tables de vérité.

Soit, par exemple, l'expression:

$Y = (A + \bar{B})\,(A + B) + C\,(\bar{A} + B)$

$= (A + \bar{B})\,A + (A + \bar{B})\,B + C\,(\bar{A} + B)$ (distributivité)

$= AA + \bar{B}A + AB + \bar{B}B + C\bar{A} + CB$ (distributivité)

or $AA = A$ (idempotence)

$B\bar{B} = 0$ (complémentarité)

d'où:

$$Y = A + \bar{B}A + AB + C\bar{A} + CB$$

Or $A + \bar{B}A = A\,(1 + \bar{B})$ (mise en facteur)

$= A\,(1)$

$= A$ (identités remarquables)

d'où

$$Y = A + AB + C\bar{A} + CB$$

or

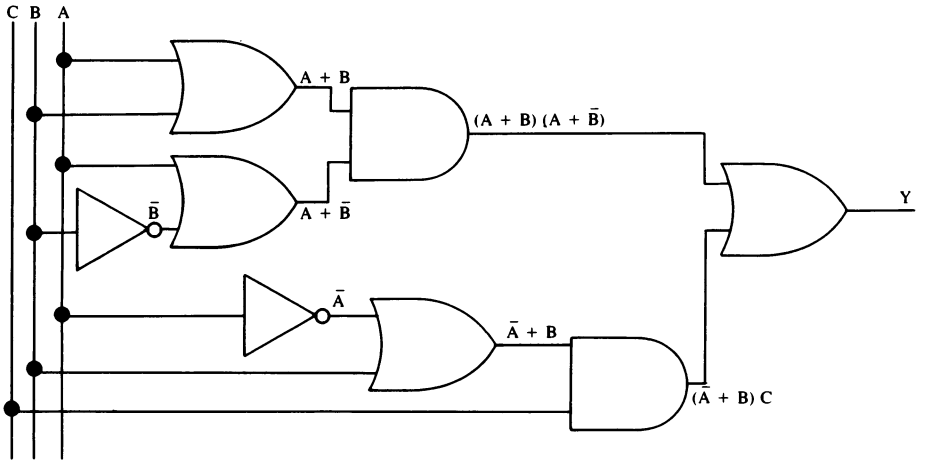
$(A + AB) = A\,(1 + B)$

$= A \cdot (1)$

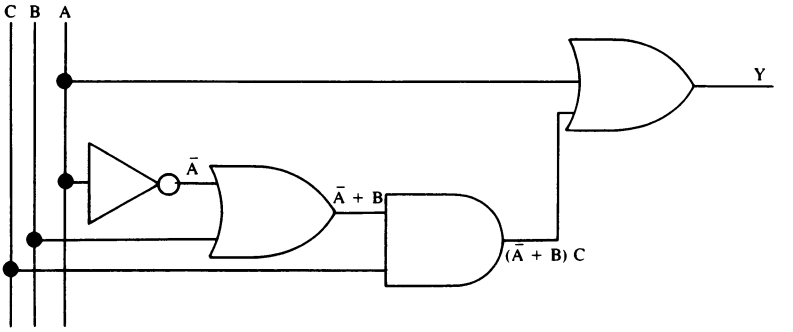
$= A$

donc $Y = A + (\bar{A} + B)\,C$ (mise en évidence)

Nous avons simplifié cette expression par déduction en nous servant de lois exposées précédemment. Ici la simplification consiste à réduire le nombre de lettres de l'expression de départ (sept lettres au départ, quatre à l'arrivée). Pour montrer l'avantage que cela représente, comparons les schémas d'implantation ci-dessous de ces deux expressions.



$Y = (A + B)(A + \bar{B}) + (\bar{A} + B)\,C$: total huit portes



$Y = A + (\bar{A} + B)\,C$: total quatre portes

La deuxième implantation économise quatre portes.

Cette expression simplifiée nous permet de dresser une table de vérité plus simple:

A	B	C	$\bar{A}$	$\bar{A}$ + B	($\bar{A}$ + B) C	A + ($\bar{A}$ + B) C
0	0	0	1	1	0	0
0	0	1	1	1	1	1
0	1	0	1	1	0	0
0	1	1	1	1	1	1
1	0	0	0	0	0	1
1	0	1	0	0	0	1
1	1	0	0	1	0	1
1	1	1	0	1	1	1

Comme en algèbre «ordinaire» les expressions de l'algèbre booléenne sont innombrables. Mais les fonctions de deux variables prenant seulement les valeurs 0 et 1 sont dénombrables. C'est ce que nous verrons maintenant.

2-7 TABLE DES FONCTIONS DE DEUX VARIABLES

Nous avons vu jusqu'à maintenant trois opérations sur les variables booléennes. D'autres opérations sont définies, nommées et utilisées en techniques numériques. Les trois opérations que nous avons étudiées sont la *négation logique* (NON), l'*addition logique* (OU) et la *multiplication logique* (ET). Chacune a été définie axiomatiquement par une table de vérité. Il y a seize fonctions possibles pour deux variables. Voici la liste des valeurs de ces seize fonctions (deux variables permettent quatre combinaisons (2^2) et ces quatre combinaisons donnent $2^4 = 16$ combinaisons différentes pour la fonction).

Pour retrouver ces seize fonctions, il suffit d'écrire tous les nombres entiers de 0 à 15 dans l'ordre binaire naturel.

x	y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Plusieurs de ces fonctions sont très simples. Par exemple F_0 est égale à 0 et F_{15} est égale à 1. Toutes ces fonctions peuvent être exprimées au moyen des opérateurs élémentaires définis précédemment.

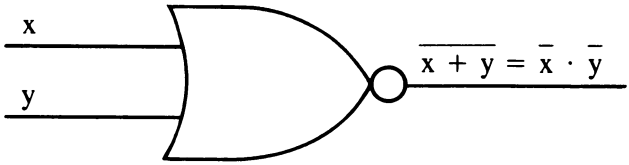
Vérifier si le tableau ci-dessous vous donne la table ci-dessus:

$F_0 = 0$	$F_4 = x\bar{y}$	$F_8 = xy$	$F_{12} = x$
$F_1 = \bar{x}\bar{y}$	$F_5 = \bar{y}$	$F_9 = \bar{x}\bar{y} + xy$	$F_{13} = x + \bar{y}$
$F_2 = \bar{x}y$	$F_6 = x\bar{y} + \bar{x}y$	$F_{10} = y$	$F_{14} = x + y$
$F_3 = \bar{x}$	$F_7 = \bar{x} + \bar{y}$	$F_{11} = \bar{x} + y$	$F_{15} = 1$

Nous connaissons déjà les fonctions F_3, F_5 (négations d'une variable), F_8 (fonction ET) et F_{14} (fonction OU).

2-7-1 AUTRES FONCTIONS TRÈS SOUVENT UTILISÉES

2-7-1-1 $F_1 = \bar{x}\,\bar{y} = \overline{x + y}$ (selon le deuxième théorème de De Morgan vu plus loin) est appelée fonction ou opération NON-OU, voici son symbole graphique:



Cette fonction est implantée par le circuit intégré 7402 représenté ci-dessous.

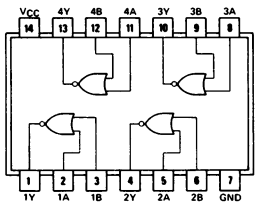
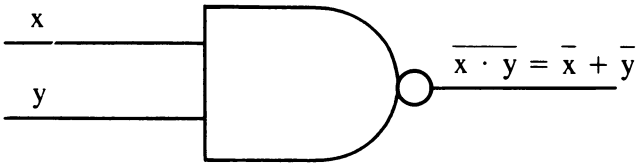


Figure 2-5 Implantation de la fonction NON-OU par le CI 7402 comprenant quatre portes NON-OU à deux entrées.

2-7-1-2 $F_7 = \bar{x} + \bar{y} = \overline{x \cdot y}$ (selon le premier théorème de De Morgan vu plus loin) est appelée fonction ou opération NON-ET, voici son symbole graphique et son circuit intégré:



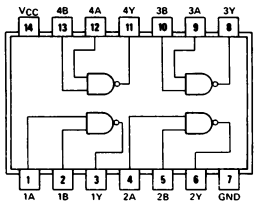
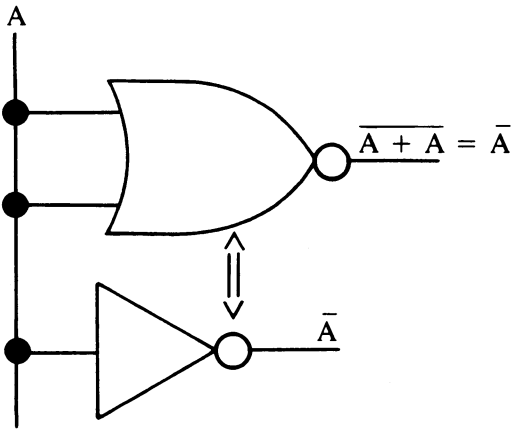


Figure 2-6 Implantation de la fonction NON-ET par le CI 7400 comprenant quatre portes NON-ET à deux entrées.

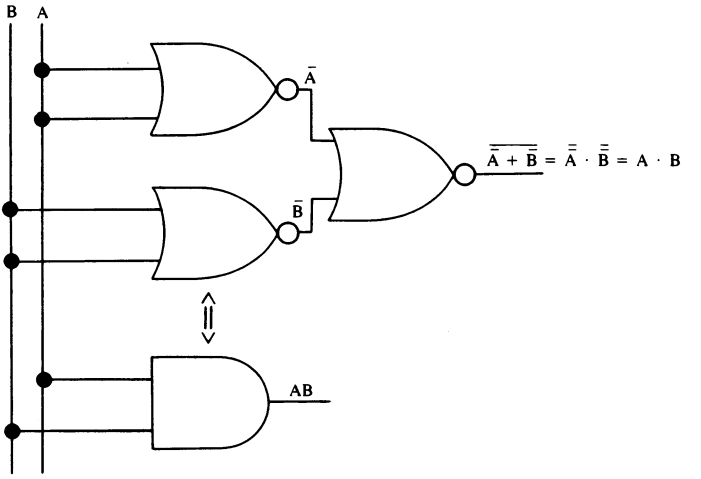
2-7-1-3 Éléments de connexion universels

Ces deux fonctions sont utilisées pour générer toutes les fonctions booléennes possibles. On dit aussi que ce sont des «éléments de connexion universels». Lorsque nous avons jeté les bases de l'algèbre de Boole, nous avons défini trois opérations: la négation, l'opération ET et l'opération OU et nous avons affirmé que toutes les fonctions booléennes étaient une combinaison de ces trois opérations. Nous allons voir les montages de portes NON-OU et NON-ET générant ces trois opérations, ces portes seront donc des *éléments de connexion universels*. Considérons tout d'abord la porte NON-OU.

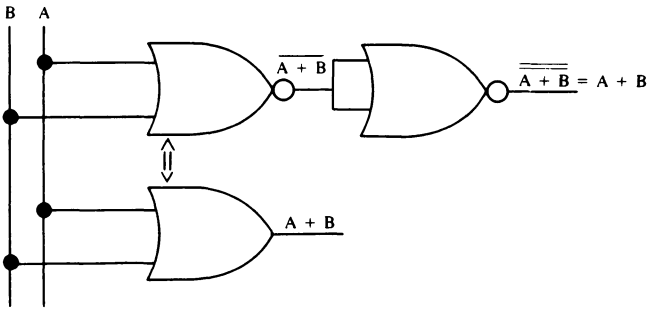
a) Négation



b) Opération ET

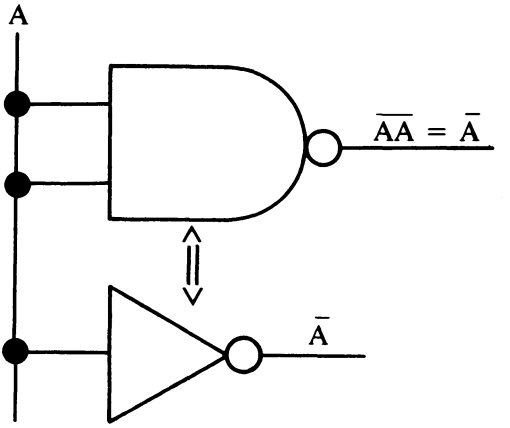


c) Opération OU

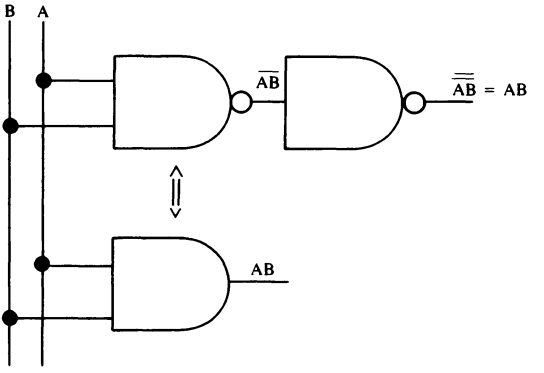


La porte NON-OU est donc un élément de connexion universel. Il en est de même de la porte NON-ET.

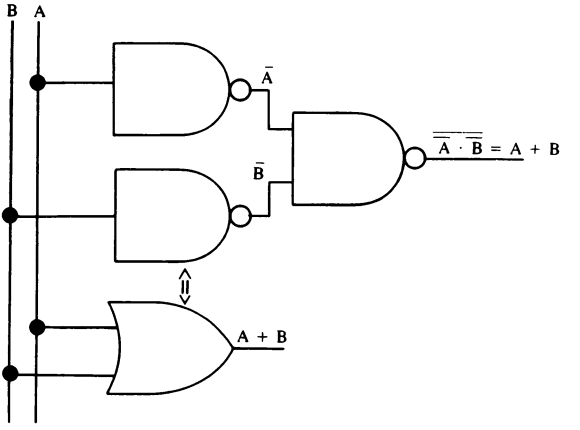
a) Négation



b) Opération ET



c) Opération OU



Un des avantages de ces éléments de connexion universels est de permettre l'implantation de n'importe quelle fonction logique à l'aide d'un seul type de porte. On peut donc en acheter une grande quantité pour bénéficier d'un prix de vente plus bas ou bien en stocker en prévision d'éventuelles pannes.

2-7-1-4 $F_6 = x\,\bar{y} + \bar{x}\,y$ est la fonction OU *exclusif*. Son symbole opératoire est $\oplus$, donc $F_6 = x \oplus y$.

L'usage quotidien de OU est ambigu, il peut signifier «l'un ou l'autre ou les deux» ou «l'un ou l'autre et non les deux».

On peut être élancé ou intelligent ou les deux mais on est à Sorel ou à Montréal et non aux deux endroits à la fois. Il faut donc définir cette opération avec rigueur. C'est pour cela qu'on l'exprime à l'aide d'un opérateur différent.

Symbole graphique du OU exclusif:

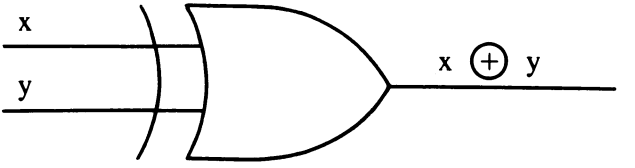
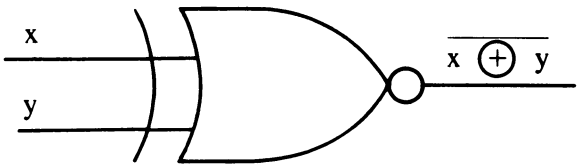


Table de vérité du OU exclusif:

Entrées		Sorties
x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

2-7-1-5 $F_9 = \bar{x}\bar{y} + x\,y$ est la fonction inverse du OU exclusif, c'est la fonction *égalité* ou *coïncidence*. Sa sortie vaut 1 si et seulement si les deux entrées sont égales.

Son symbole graphique est:



Ces deux fonctions sont implantées par des circuits intégrés: le 7486 pour le OU exclusif et le 74266 pour la fonction égalité.

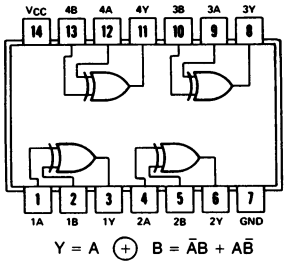


Figure 2-7 Implantation de la fonction OU exclusif par le CI 7486 comportant quatre portes OU exclusif à deux entrées.

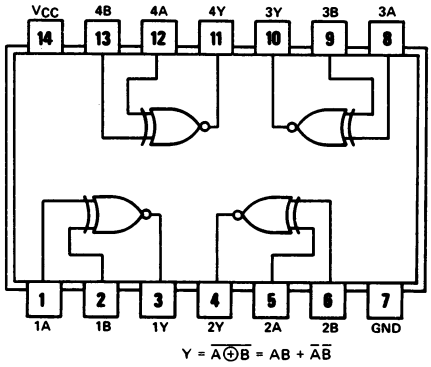


Figure 2-8 Implantation de la fonction égalité par le CI 74266 comprenant quatre portes NON-OU exclusif à deux entrées.

REMARQUES

1) $F_a = \overline{F_{15-a}}$ avec $a \in \{0, 1, 2, \ldots, 7\}$:

$F_0 = \overline{F_{15}}$ $\quad F_1 = \overline{F_{14}}$

$F_2 = \overline{F_{13}}$ $\quad$ etc... $\quad F_7 = \overline{F_8}$

2) On ne peut pratiquement dresser le tableau de toutes les fonctions possibles pour n'importe quel nombre de variables. Pour trois variables, par exemple, on a $2^3 = 8$ combinaisons possibles et donc $2^8 = 256$ fonctions. Il nous faut donc d'autres méthodes pour étudier ces fonctions à trois, quatre, cinq variables.

2-8 RELATIONS DE BASE DE L'ALGÈBRE BOOLÉENNE

Nous allons démontrer un certain nombre de relations de base de l'algèbre de Boole. Ces relations serviront de références pour simplifier des expressions booléennes ou pour démontrer de nouvelles relations.

Ces démonstrations ou preuves s'appuient uniquement sur des expressions prouvées elles-aussi ou acceptées à titre d'axiomes. À partir de la définition des variables booléennes et des trois opérations élémentaires nous échafauderons petit à petit une algèbre qui nous servira à mettre en équation un problème technique relevant de l'algèbre logique, à le résoudre et à implanter sa solution par le circuit le plus simple possible. Les applications figurent au chapitre 3.

Le schéma d'implantation de chacune de ces relations permettra de constater leur puissance de simplification de visu.

Les relations de base:

I) $x\,y + x\,\bar{y} = x$	I') $(x + y)\,(x + \bar{y}) = x$

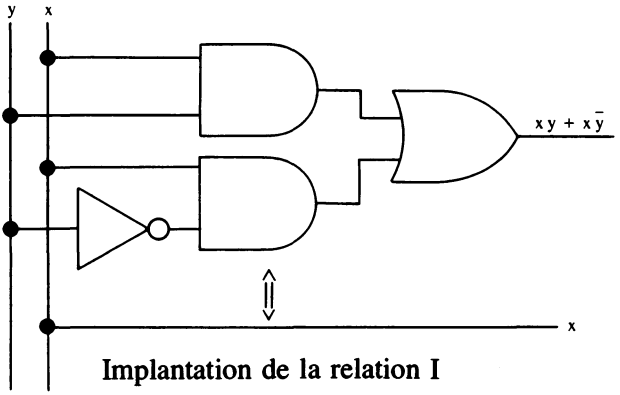
Preuves

I) $x\,y + x\,\bar{y} = x$

$= x\,(y + \bar{y})$ (mise en facteur)

$= x \cdot 1$ car $y + \bar{y} = 1$ (complémentarité)

$= x$ car $x \cdot 1 = 1 \cdot x = x$
(commutativité et identité remarquable)



Implantation de la relation I

I') $(x + y)\,(x + \bar{y}) = x$

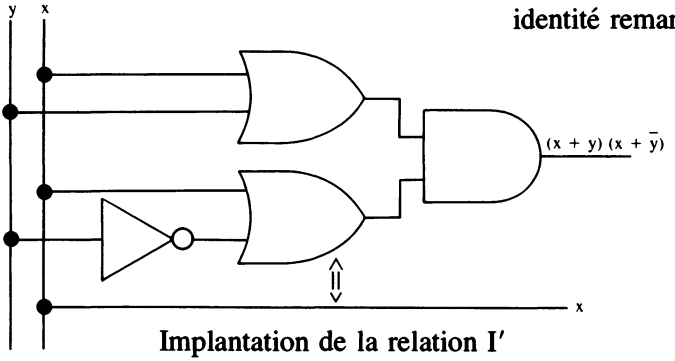
$= x\,x + x\,\bar{y} + y\,x + y\,\bar{y}$ (distributivité)

$= x + x\,\bar{y} + y\,x$ car $x\,x = x$ (idempotence)
et $y\,\bar{y} = 0$ (complémentarité)

$= x\,(1 + \bar{y} + y)$ (mise en facteur)

$= x\,(1 + 1)$ car $\bar{y} + y = 1$ (complémentarité)

$= x$ car $1 + 1 = 1$ (idempotence)
$x \cdot 1 = 1 \cdot x = x$ (commutativité et identité remarquable)

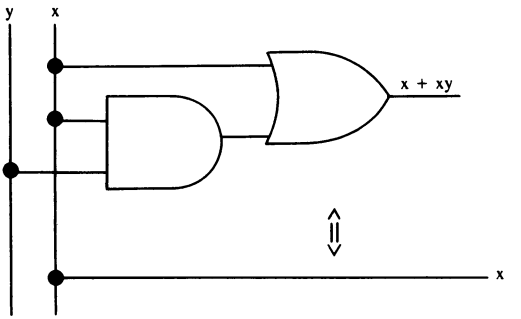


Implantation de la relation I'

II) $x + xy = x$	II′) $x(x + y) = x$

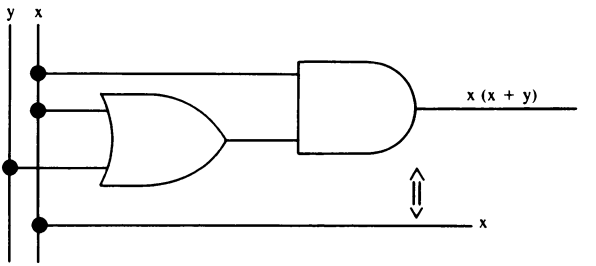
Preuves

II) $x + xy = x$
$= x(1 + y)$ (mise en facteur)
$= x \cdot 1$ car $1 + y = 1$ (identité remarquable)
$= x$ car $x \cdot 1 = 1 \cdot x = x$ (commutativité et identité remarquable)



Implantation de la relation II

II′) $x(x + y) = x$
$= xx + xy$ (distributivité)
$= x + xy$ car $xx = x$ (idempotence)
$= x$ selon II)



Implantation de la relation II′

III) $x + \bar{x}y = x + y$	III′) $x(\bar{x} + y) = xy$

Preuves

III) $x + \bar{x}y = x + y$

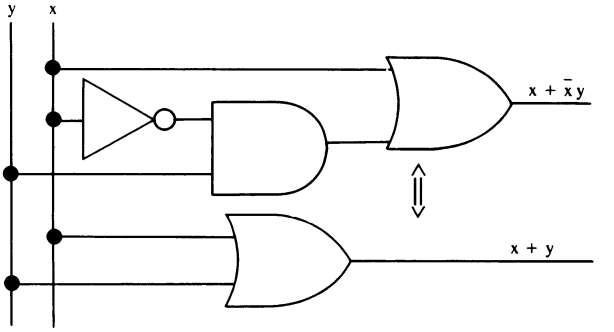
$= x + xy + \bar{x}y$ car $x + xy = x$ selon II)

$= x + y(x + \bar{x})$ (mise en facteur)

$= x + y \cdot 1$ car $x + \bar{x} = 1$ (complémentarité)

$= x + y$ car $y \cdot 1 = 1 \cdot y = y$

(commutativité et identité remarquable)



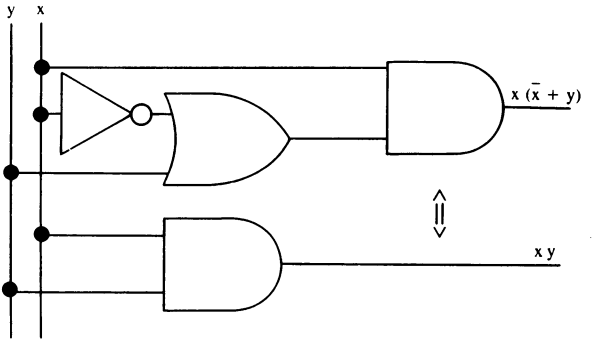
Implantation de la relation III

III′) $x(\bar{x} + y) = xy$

$= x\bar{x} + xy$ (distributivité)

$= 0 + xy$ car $x\bar{x} = 0$ (complémentarité)

$= xy$ car $0 + xy = xy$ (identité remarquable)



Implantation de la relation III′

IV) $x\,y + \bar{x}\,z + y\,z = x\,y + \bar{x}\,z$

IV′) $(x + y)\,(\bar{x} + z)\,(y + z) = (x + y)\,(\bar{x} + z)$

Preuves

IV) $xy + \bar{x}z + yz = xy + \bar{x}z$

$= xy + (y + \bar{x})\,z$ (mise en facteur)

$= xy + (y + \bar{y}\,\bar{x})\,z$

car $y + \bar{x} = y + \bar{y}\,\bar{x}$ selon III)

$= xy + yz + \bar{y}\,\bar{x}\,z$ (distributivité)

$= (x + z)\,y + \bar{y}\,\bar{x}\,z$ (mise en facteur)

$= (x + \bar{x}\,z)\,y + \bar{y}\,\bar{x}\,z$ car $x + z = x + \bar{x}\,z$ selon III)

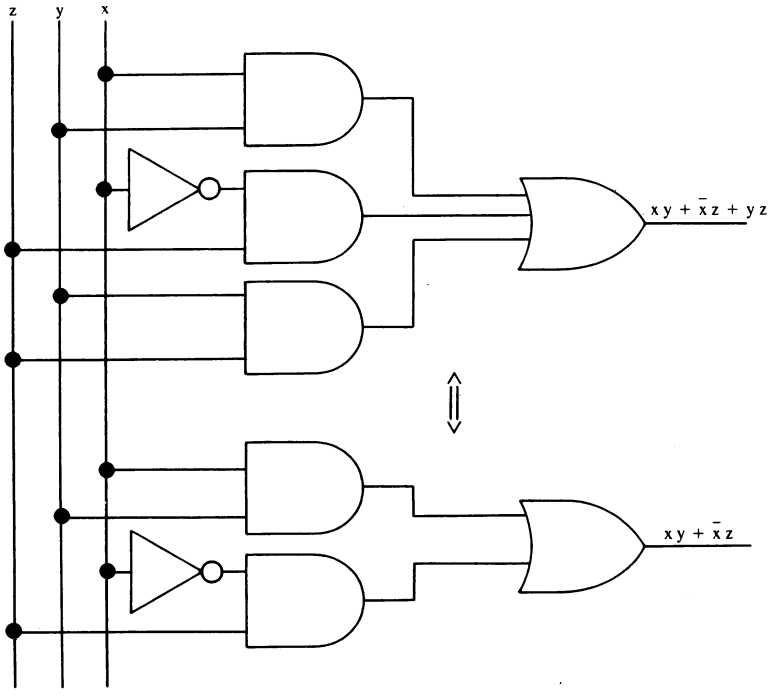
$= xy + \bar{x}zy + \bar{y}\,\bar{x}\,z$ (distributivité)

$= xy + \bar{x}\,z\,(y + \bar{y})$ (mise en facteur)

$= xy + \bar{x}\,z \cdot 1$ car $y + \bar{y} = 1$ (complémentarité)

$= xy + \bar{x}\,z$ car $\bar{x}\,z \cdot 1 = 1 \cdot \bar{x}\,z = \bar{x}\,z$

(commutativité et identité remarquable)



Implantation de la relation IV

IV′) $(x + y)(\bar{x} + z)(y + z) = (x + y)(\bar{x} + z)$

$= (x + y)(\bar{x}y + \bar{x}z + zy + zz)$
(distributivité)

$= (x + y)(\bar{x}y + \bar{x}z + zy + z)$
car $zz = z$ (idempotence)

$= (x + y)\left[(\bar{x}y + \bar{x}z + z(y + 1)\right]$
(mise en facteur)

$= (x + y)(\bar{x}y + \bar{x}z + z)$ car $y + 1 = 1$
(identité remarquable)
et $z \cdot 1 = 1 \cdot z = z$
(commutativité et identité remarquable)

$= (x + y)\left[\bar{x}y + (\bar{x} + 1)z\right]$
(mise en facteur)

$= (x + y)(\bar{x}y + z)$ (voir plus haut)

$= x\bar{x}y + xz + \bar{x}yy + yz$
(distributivité)

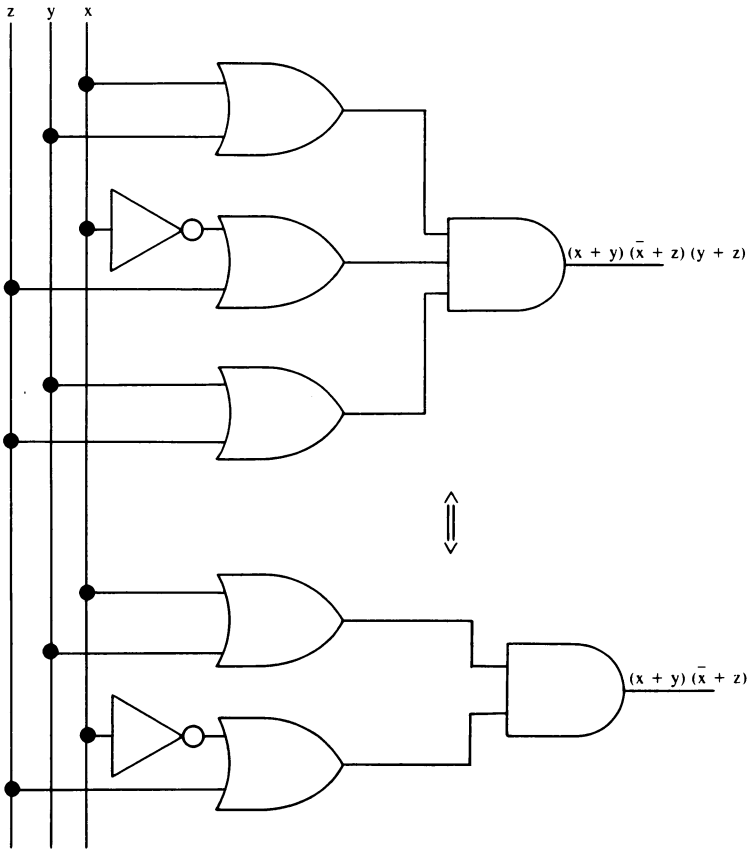
$= xz + \bar{x}y + yz$
car $x\bar{x}y = 0y = 0$ (complémentarité
et identité remarquable)
et $yy = y$ (idempotence)

$= 0 + xz + \bar{x}y + yz$
(identité remarquable)

$= x\bar{x} + xz + \bar{x}y + y\underline{z}$
car $0 = x\bar{x}$ (commutativité
et complémentarité)

$= x(\bar{x} + z) + y(\bar{x} + z)$
(mise en facteur)

$= (x + y)(\bar{x} + z)$ (mise en facteur)



Implantation de IV′

V) $x\,y + x\,\bar{y}\,z = x\,y + x\,z$
V') $(x + y)\,(x + \bar{y} + z) = (x + y)\,(x + z)$

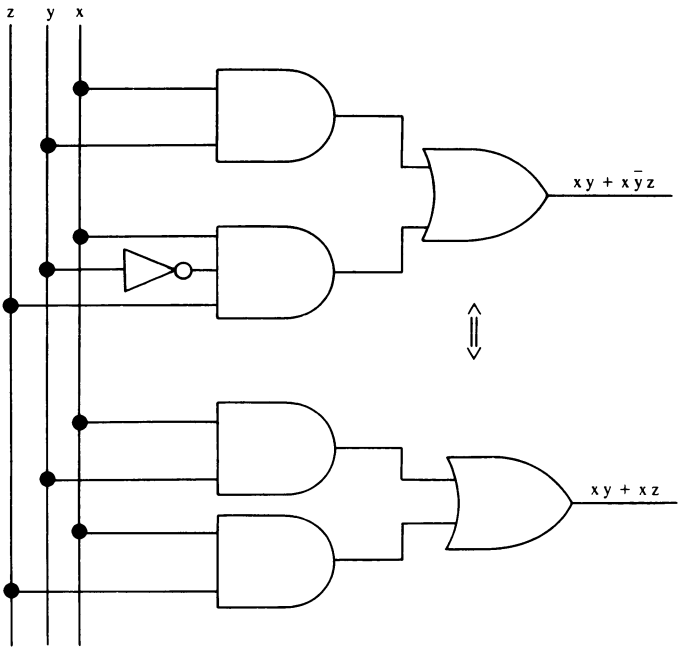
Preuves

V) $x\,y + x\,\bar{y}\,z = x\,y + x\,z$

$= x\,(y + \bar{y}\,z)$ (mise en facteur)

$= x\,(y + z)$ car $y + \bar{y}\,z = y + z$ selon III)

$= x\,y + x\,z$ (distributivité)



Implantation de la relation V

V′) $(x + y)(x + \bar{y} + z) = (x + y)(x + z)$

$= x\,x + x\,\bar{y} + x\,z + y\,x + y\,\bar{y} + y\,z$
(distributivité)

$= x + x\,\bar{y} + x\,z + y\,x + y\,z$
car $x\,x = x$ (idempotence)
et $y\,\bar{y} = 0$ (complémentarité)

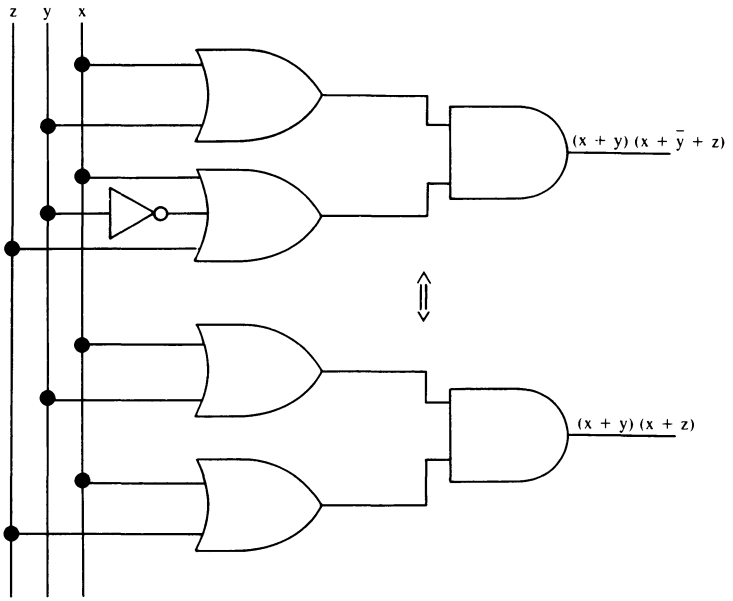
$= x\,(1 + \bar{y}) + x\,z + y\,x + y\,z$
(mise en facteur)

$= x + x\,z + y\,x + y\,z$
car $1 + \bar{y} = 1$ (identité remarquable)
et $x \cdot 1 = 1 \cdot x = x$
(commutativité et identité remarquable)

$= x\,x + x\,z + y\,x + y\,z$ (idempotence)

$= x\,(x + z) + y\,(x + z)$ (mise en facteur)

$= (x + y)(x + z)$ (mise en facteur)



Implantation de la relation V′

2-9 THÉORÈMES DE DE MORGAN

THÉORÈME 1 La négation d'un produit de variables est égale à la somme des négations des variables.

$$\overline{x\,y\,z} = \bar{x} + \bar{y} + \bar{z}$$

Preuve par induction parfaite

x	y	z	xyz	$\overline{xyz}$	$\bar{x} + \bar{y} + \bar{z}$
0	0	0	0	1	1
0	0	1	0	1	1
0	1	0	0	1	1
0	1	1	0	1	1
1	0	0	0	1	1
1	0	1	0	1	1
1	1	0	0	1	1
1	1	1	1	0	0

THÉORÈME 2 La négation d'une somme de variables est égale au produit des négations des variables.

$$\overline{x + y + z} = \bar{x}\,\bar{y}\,\bar{z}$$

Preuve par induction parfaite

x	y	z	x + y + z	$\overline{x + y + z}$	$\bar{x}\,\bar{y}\,\bar{z}$
0	0	0	0	1	1
0	0	1	1	0	0
0	1	0	1	0	0
0	1	1	1	0	0
1	0	0	1	0	0
1	0	1	1	0	0
1	1	0	1	0	0
1	1	1	1	0	0

REMARQUE On a déjà vu quelque chose de semblable lors de l'étude des circuits NON-ET et NON-OU.

On peut généraliser ces résultats à n'importe quel nombre de variables.

Ces deux théorèmes sont très utiles pour les circuits logiques. Ils permettent entres autres de transformer un produit de sommes ou une somme de produits. Prenons un exemple:

Soit l'expression $Z = (x + \bar{y} + z)\,(x + \bar{z})\,(\bar{x} + \bar{y})$,
produit de trois sommes, à transformer.

On peut écrire $Z = \bar{\bar{Z}}$ puisqu'une double inversion est une opération d'effet nul, d'où:

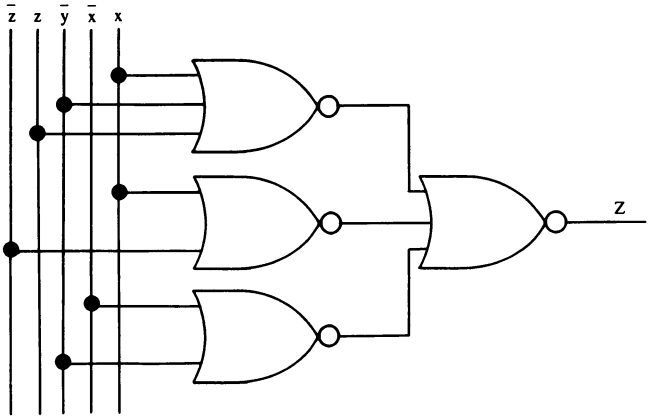
$$Z = \bar{\bar{Z}}$$

$$= \overline{(x + \bar{y} + z)\,(x + \bar{z})\,(\bar{x} + \bar{y})}$$

$$= \overline{\bar{x}\bar{y}\bar{z}} + \overline{\bar{x}\bar{z}} + \overline{\bar{x}\bar{y}}$$

D'après le théorème 1 de De Morgan, on remarquera que l'expression obtenue est une implantation différente de la même fonction.

En supposant que les variables et leur complément sont disponibles, l'expression originale nécessite, pour être implantée, trois portes OU $(x + \bar{y} + z)$; $(x + \bar{z})$; $(\bar{x} + \bar{y})$ et une porte ET à trois entrées pour faire le produit des trois sorties des portes OU. Par contre, l'expression finale nécessite quatre portes NON-OU suivant le schéma ci-dessous:



REMARQUE: L'expression ci-dessus n'est pas simplifiée.

Prenons $Z = \overline{\overline{x+\bar{y}+z} + \overline{x+\bar{z}} + \overline{\bar{x}+\bar{y}}}$

D'après le théorème 2 de De Morgan, on obtient:

$$Z = \overline{\bar{x}\,y\,\bar{z} + \bar{x}\,z + x\,y}$$

$$= \overline{\bar{x}\,(z + \bar{z}\,y) + x\,y}$$

$$= \overline{\bar{x}\,(z + y) + x\,y} \quad \text{d'après la relation de base III)}$$

$$= \overline{\bar{x}\,z + \bar{x}\,y + x\,y}$$

$$= \overline{\bar{x}\,z + y\,(\bar{x} + x)}$$

$$= \overline{\bar{x}\,z + y}$$

On a alors une expression simplifiée qui peut aussi s'écrire:

$$Z = \overline{\bar{x}\,z}\ \ \bar{y}$$

$$= (x + \bar{z})\ \bar{y}$$

$$= \bar{y}\,x + \bar{y}\,\bar{z}$$

$$= \bar{y}\,(x + \bar{z}): \quad \text{implantation ET, OU.}$$

On peut aussi écrire:

$Z = \bar{\bar{Z}}$ puisqu'une double inversion est une opération d'effet nul.

$$= \overline{\bar{y}\,x + \bar{y}\,z}$$

$$= \overline{\overline{\bar{y}\,x}\ \ \overline{\bar{y}\,z}}$$

$\bar{y}\,x$ $\quad$ $\bar{y}\,z$: implantation NON-ET.

De la même façon on obtient:

$$Z = \overline{\bar{y}\,(x + z)}$$

$$= \overline{y + \overline{(x + \bar{z})}}: \text{ implantation NON-OU.}$$

En résumé, retenons les trois formes de la fonction Z:

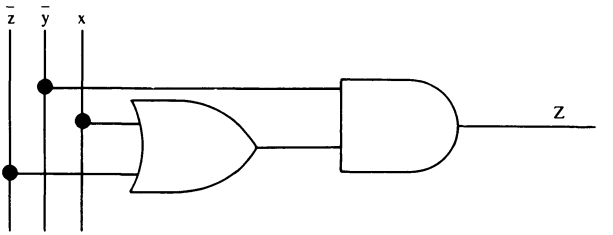
a) $Z = \bar{y}\,(x + \bar{z})$: forme ET, OU.

b) $Z = \overline{\bar{y}\,x}\ \ \overline{\bar{y}\,z}$: forme NON-ET.

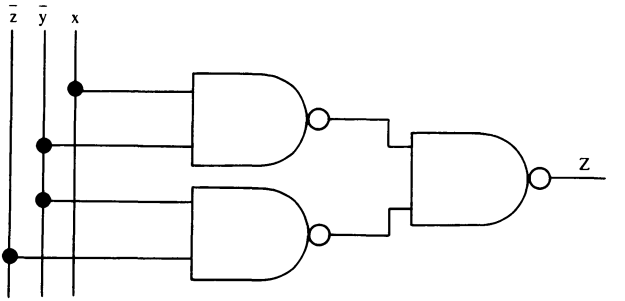
c) $Z = \overline{y + \overline{(x + \bar{z})}}$: forme NON-OU.

Elles ont chacune une implantation différente.

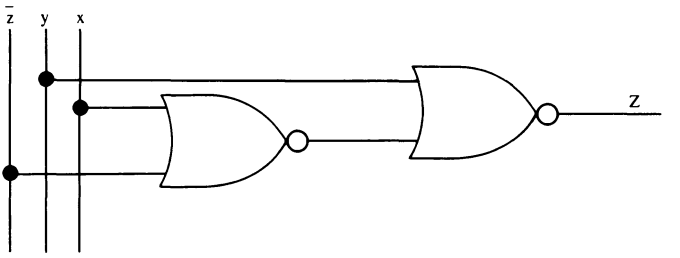
a) Implantation ET, OU



b) Implantation NON-ET



c) Implantation NON-OU



Nous voyons donc qu'en manipulant algébriquement une fonction on peut choisir le type de portes de son implantation. Dans notre cas, l'implantation NON-OU est la plus simple puisqu'elle ne nécessite qu'un seul type de porte (donc un seul type de circuit intégré) et seulement deux portes (donc un seul circuit intégré). C'est un exemple de l'utilité pratique des théorèmes de De Morgan.

2-10 DUALITÉ DE L'ALGÈBRE DE BOOLE

En se reportant aux deux paragraphes précédents, on remarque que toutes les fonctions booléennes et les théorèmes de De Morgan vont toujours par deux. Le remplacement dans la première relation des opérations $(\cdot)$ par $(+)$ et $(+)$ par $(\cdot)$ donne la deuxième.

Remarquons que les lois des paragraphes 2-5-5, 6 et 7 se transforment entre elles lorsqu'on remplace 1 par 0, + par $\cdot$ et vice versa.

$1 + A = 1 \qquad 0 \cdot A = 0 \qquad 1 \cdot A = A \qquad 0 + A = A$

$A + A = A \qquad A \cdot A = A \qquad A + \bar{A} = 1 \qquad A \cdot \bar{A} = 0$

Cette propriété générale est appelée *dualité de l'algèbre de Boole.*

Si donc on démontre une relation on peut écrire immédiatement sa duale en remplaçant les opérations $(\cdot)$ par $(+)$ et $(+)$ par $(\cdot)$, 1 par 0 et 0 par 1.

L'obtention d'une première forme simplifiée d'une expression permet d'écrire immédiatement une autre égalité, duale de la première.

Soit l'expression $Z = a\bar{b} + \bar{a}\bar{b} + \bar{a}b$

$= \bar{b}(a + \bar{a}) + \bar{a}b$ (mise en facteur)

$= \bar{b} + \bar{a}b$ puisque $a + \bar{a} = 1$ et $\bar{b} \cdot 1 = \bar{b}$

$= \bar{a} + \bar{b}$ (relation de base III et commutativité)

D'où: $a\bar{b} + \bar{a}\bar{b} + \bar{a}b = \bar{a} + \bar{b}$ (1)

On en déduit donc immédiatement la relation:

$$(a + \bar{b})(\bar{a} + \bar{b})(\bar{a} + b) = \bar{a} \cdot \bar{b} \quad (2)$$

en se servant de la dualité de l'algèbre de Boole.

Vérification Développons le premier membre de (2), on aura successivement:

$$(a\bar{a} + a\bar{b} + \bar{a}\bar{b} + \bar{b})(\bar{a} + b) = [0 + \bar{b}(1 + a + \bar{a})](\bar{a} + b)$$
$$= \bar{b}(\bar{a} + b)$$
$$= \bar{b}\bar{a} + \bar{b}b$$
$$= \bar{a}\bar{b}, \text{ soit le deuxième membre de (2).}$$

RÉSUMÉ

$xy + x\bar{y} = x$	donne	$(x + y)(x + \bar{y}) = x$
$x + xy = x$	"	$x(x + y) = x$
$x + \bar{x}y = x + y$	"	$x(\bar{x} + y) = xy$
$xy + \bar{x}z + yz = xy + \bar{x}z$	"	$(x + y)(\bar{x} + z)(y + z) = (x + y)(\bar{x} + z)$
$xy + x\bar{y}z = xy + xz$	"	$(x + y)(x + \bar{y} + z) = (x + y)(x + z)$
$\overline{xyz} = \bar{x} + \bar{y} + \bar{z}$	"	$\overline{x + y + z} = \bar{x}\bar{y}\bar{z}$

2-11 SIMPLIFICATION ALGÉBRIQUE DES ÉQUATIONS BOOLÉENNES

Pour simplifier algébriquement une fonction booléenne: la développer, effectuer des mises en facteur et simplifier, selon les lois fondamentales et les relations démontrées. Simplifier une fonction revient donc à l'écrire à l'aide d'un nombre minimum de termes.

Exemple 1 Simplifier $z = (a + b)(\bar{b} + c)(\bar{a} + c)$

Solution

Développons, on aura:

$$z = (a\bar{b} + ac + b\bar{b} + bc)(\bar{a} + c)$$

Or, $b\bar{b} = 0$, d'où:

$z = (a\bar{b} + ac + bc)(\bar{a} + c)$ développons, on aura:

$z = a\bar{b}\bar{a} + a\bar{b}c + ac\bar{a} + acc + bc\bar{a} + bcc$

Or, $\bar{a}a = 0$, d'où:

$\bar{a}a\bar{b} = 0$ et $\bar{a}ac = 0$

Mais $cc = c$, d'où l'on tire:

$z = \bar{a}bc + a\bar{b}c + ac + bc$

a est en facteur dans les 2e et 3e termes, b dans les 1er et 4e et c dans chacun, d'où:

$z = c\left[a(\bar{b} + 1) + b(\bar{a} + 1)\right]$

Or, $\bar{a} + 1 = \bar{b} + 1 = 1$, d'où:

$z = (a + b)c$

Exemple 2 Simplifier $z = (a + b)(\bar{a} + \bar{b} + \bar{c})(a + c)$

Solution

Développons, on aura:

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

Or, $a\bar{a} = 0$ et $b\bar{b} = 0$, d'où

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

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

Or, $aa = a$, $a\bar{a} = 0$ et $c\bar{c} = 0$, d'où:

$z = a\bar{b} + a\bar{b}c + a\bar{c} + b\bar{a}c + b\bar{c}a$

$= a\bar{b}(1 + c) + a\bar{c}(1 + b) + b\bar{a}c$

$= a\bar{b} + a\bar{c} + \bar{a}bc$

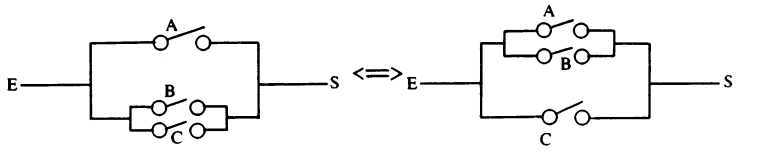
$= a(\bar{b} + \bar{c}) + \bar{a}bc$

$= a\overline{bc} + \bar{a}bc$, soit

$= a \oplus bc$

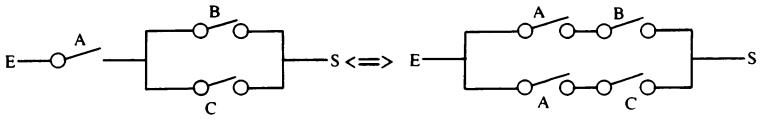
PROBLÈMES

2-1 a) Mettre les circuits suivants en équation.

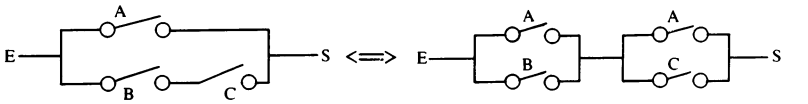


b) Écrire et nommer la loi de l'algèbre Boole établissant leur équivalence.

2-2 Reprendre le problème 2-1 pour les deux circuits suivants:



2-3 Reprendre le problème 2-1 pour les deux circuits suivants:



2-4 Construire les circuits équivalents représentant respectivement les deux membres des équations suivantes:

a) $1 \cdot A = A$; b) $1 + A = 1$; c) $A + A = A$;

d) $0 + A = A$; e) $0 \cdot A = 0$; f) $A \cdot A = A$.

Développer les expressions E suivantes, les simplifier algébriquement, si possible, et dresser leur table de vérité.

2-5 $E = (a + d)(a\,b + a\,c)(\bar{a}\,\bar{c} + b)$

2-6 $E = (a + c)(b + d)$

2-7 $E = (a + c + d)(b + c + d)$

2-8 $E = (a\,\bar{b} + c + c\,d)(\bar{a}\,\bar{c} + b\,c + d)$

2-9 $E = (a\,\bar{b} + a\,b + a\,c)(\bar{a}\,\bar{b} + a\,b + a\,\bar{c})$

Simplifier, si possible, les expressions E suivantes:

2-10 $E = x\,y\,z + x\,\bar{y}\,\bar{z}$

2-11 $E = a\,(b\,\bar{c} + \bar{b}\,c)$

2-12 $E = a\bar{b} + ab$

2-13 $E = ab(a\bar{b}c + a\overline{bc} + ab\bar{c})$

2-14 $E = abc + a\overline{bc} + \bar{a}\bar{b}\bar{c}$

2-15 $E = (a + b)(a + c)(\bar{a} + \bar{b})$

2-16 $E = a\bar{b} + b\bar{c}$

2-17 $E = \bar{a}bc + a\bar{b}c + ab\bar{c}$

2-18 $E = a\overline{bc} + \bar{a}\bar{b}\bar{c} + \bar{a}b\bar{c} + \overline{ab}c$

2-19 $E = abc + \bar{a}bc + a\bar{b}c + ab\bar{c} + a\overline{bc} + \bar{a}b\bar{c} + \bar{a}\bar{b}c$

2-20 $E = a(a + b + c)(\bar{a} + b + c)(a + \bar{b} + c)(a + b + \bar{c})$

2-21 $E = (a + b + c)(a + \bar{b} + \bar{c})(a + b + \bar{c})(a + \bar{b} + c)$

Former le complément $\bar{E}$ des expressions E suivantes:

2-22 $E = a + bc + ab$

2-23 $E = (a + b)(b + c)(a + c)$

2-24 $E = ab + \bar{b}c + c\bar{d}$

2-25 $E = a(b + c)(\bar{c} + \bar{d})$

2-26 $E = ab(\bar{c}d + \bar{b}c)$

Écrire l'égalité duale D des égalités E ci-dessous:

2-27 $E = a + bc + ab = a + bc$

2-28 $E = a(b + c)(\bar{c} + \bar{d}) = (ab\bar{c} + ab\bar{d} + ac\bar{d}$

2-29 $E = ab(\bar{c}d + \bar{b}c) = ab\bar{c}d$

3

Représentation, simplification, implantation des fonctions logiques

3-1 OBJECTIFS

1. Savoir représenter une fonction logique sous quatre formes:
   1) expression algébrique
   2) table de vérité
   3) table de Karnaugh
   4) logigramme
2. Savoir extraire deux formes canoniques d'une table de vérité et d'une table de Karnaugh. Déduire les deux autres formes des deux premières.
3. Savoir simplifier une expression booléenne à partir des tables de Karnaugh complètes ou non.
4. Savoir implanter une fonction logique avec les circuits intégrés TTL normalisés. (74XX)

3-3 MODES DE REPRÉSENTATION DES FONCTIONS LOGIQUES

3-2-1 ÉCRITURE ALGÉBRIQUE

Soit les fonctions:

$$S = x\bar{y}\bar{c} + \bar{x}y\bar{c} + \bar{x}\bar{y}c + xyc$$

$$C = xy\bar{c} + x\bar{y}c + \bar{x}yc + xyc$$

$$= xy + xc + yc$$

Ces expressions sont sous forme algébrique. Les deux expressions de C sont équivalentes. La deuxième est sous forme simplifiée. En effet:

$$C = xy\bar{c} + x\bar{y}c + \bar{x}yc + xyc$$

$$= xy(\bar{c} + c) + x\bar{y}c + \bar{x}yc$$

$$= xy + x\bar{y}c + \bar{x}yc$$

$$= x(y + \bar{y}c) + \bar{x}yc$$

$$= x(y + c) + \bar{x}yc$$

$$= xy + xc + \bar{x}yc$$

$$= xc + y(x + \bar{x}c)$$

$$= x\,c + y\,(x + c)$$

$$C = x\,y + x\,c + y\,c$$

3-2-2 TABLE DE VÉRITÉ

On peut représenter ces deux fonctions sous forme d'une table de vérité. Les entrées sont dans l'ordre binaire naturel. Nous avons trois variables x, y et c, donc $2^3 = 8$ combinaisons possibles.

c	y	x	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Exemple Représenter sous forme de table de vérité l'expression suivante:

$$z = \overline{a\,b\,(\bar{c}\,d + \bar{b}\,c)}$$

Solution Nous avons quatre variables a, b, c et d, donc $2^4 = 16$ combinaisons possibles de ces variables. D'où la table:

d	c	b	a	ab	$\bar{c}d$	$\bar{b}c$	$\bar{c}d + \bar{b}c$	$ab\,(\bar{c}d + \bar{b}c)$	z
0	0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1
0	0	1	1	1	0	0	0	0	1
0	1	0	0	0	0	1	1	0	1
0	1	0	1	0	0	1	1	0	1
0	1	1	0	0	0	0	0	0	1
0	1	1	1	1	0	0	0	0	1
1	0	0	0	0	1	0	1	0	1
1	0	0	1	0	1	0	1	0	1
1	0	1	0	0	1	0	1	0	1
1	0	1	1	1	1	0	1	1	0
1	1	0	0	0	0	1	1	0	1
1	1	0	1	0	0	1	1	0	1
1	1	1	0	0	0	0	0	0	1
1	1	1	1	1	0	0	0	0	1

Sous forme algébrique simplifiée, on a:

$$z = \overline{a\,b\,(\bar{c}\,d + \bar{b}\,c)}$$

$$= \overline{a\,b\,\bar{c}\,d + a\,b\,\bar{b}\,c}$$

$$= \overline{a\,b\,\bar{c}\,d}$$

3-2-3 TABLE DE KARNAUGH

La table de vérité présente un inconvénient: le nombre de lignes croît en même temps que le nombre de variables. Si n est le nombre de variables, le nombre de lignes est 2^n. Pour trois variables on a huit lignes, pour quatre variables seize lignes, etc. On contourne cette difficulté en plaçant les variables d'entrée aux entrées d'une *table* aussi carrée que possible. On a donc une table dite *de Karnaugh* à double entrée: une pour les *lignes* et une pour les *colonnes*. Soit n variables, la table sera la plus carrée possible pour p et q entiers tels que $p + q = n$, lorsque $p = q = n/2$ pour n pair, ou $|p-q| = 1$ pour n impair.

La table comportera alors 2^p colonnes et 2^q lignes.

On numérote ensuite les colonnes et les lignes selon le code binaire réfléchi. (On sait que, dans le code, deux nombres adjacents, c'est-à-dire l'un à la suite de l'autre, ne sont différents que par un seul bit. On se servira de cette propriété d'adjacence du code Gray pour les simplifications au paragraphe 3-4-2.) Considérons la fonction S du paragraphe 3-2-1 à trois variables, prenons $p = 2$ et $q = 1$.

On a donc $2^2 = 4$ colonnes et $2^1 = 2$ lignes, on placera donc deux variables dans les colonnes et une dans les lignes.

À partir de la table de vérité, on inscrit dans les *cases* les 0 et les 1 de la fonction S de la façon suivante. La première case en haut à gauche correspond à $x = y = c = 0$, on inscrit donc 0 dans cette case. La case intersection de la 1 ere ligne et de la 2e colonne correspond à $x = 1$, $y = c = 0$ on inscrit donc 1 dans cette case et ainsi de suite.

Table de S

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

S

En procédant de la même façon pour C, on aura:

Table de C

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

C

Pour obtenir la fonction de chaque case il suffit d'effectuer pour chaque case le produit des variables en complémentant chaque variable de valeur 0. Ce qui donne pour S:

c \ yx	00	01	11	10
0	$\bar{x}\,\bar{y}\,\bar{c}$	$x\,\bar{y}\,\bar{c}$	$x\,y\,\bar{c}$	$\bar{x}\,y\,\bar{c}$
1	$\bar{x}\,\bar{y}\,c$	$x\,\bar{y}\,c$	$x\,y\,c$	$\bar{x}\,y\,c$

S

Exemple 1 Représenter sous forme d'une table de Karnaugh l'expression $z = a\,b\,\bar{d} + \bar{a}\,\bar{b}\,c\,d + a\,b\,c + \bar{a}\,b\,c\,d + \bar{a}\,b\,c\,\bar{d}$

Solution Nous avons quatre variables, donc $n = 4$ et $p = q = n/2 = 2$. La table aura donc $2^2 = 4$ lignes et $2^2 = 4$ colonnes. Nous allons assigner les variables suivant la table ci-dessous. Pour chaque produit, inscrire 1 aux intersections des 1 de ses facteurs. Ceci étant fait, compléter avec des 0. On trouvera ci-dessous les 1 pour les facteurs des colonnes et ceux des lignes. Il suffira d'inscrire 1 à l'intersection des colonnes et des lignes indiquées.

dc \ ba	00	01	11	10
00	1	0	0	0
01	1	0	1	0
11	0	0	1	1
10	1	0	0	1

Z

$a\,b\,\bar{d}$	$a\,b$:	1re colonne	Inscrire 1 dans les cases
	$\bar{d}$:	1re et 2e lignes	de cette intersection
$\bar{a}\,\bar{b}\,c\,d$	$\bar{a}\,\bar{b}$:	1re colonne	Inscrire 1 dans la case
	$c\,d$:	dernière ligne	de cette intersection

a b c	a b:	3e colonne	Inscrire 1 dans les cases
	c:	2e et 3e lignes	de cette intersection
$\bar{a}$ b c d	$\bar{a}$ b:	dernière colonne	Inscrire 1 dans la case
	c d:	3e ligne	de cette intersection
$\bar{a}$ b $\bar{c}$ d	$\bar{a}$ b:	dernière colonne	Inscrire 1 dans la case
	$\bar{c}$ d:	dernière ligne	de cette intersection

Compléter la table de Karnaugh avec des 0.

Exemple 2 Représenter sous forme d'une table de Karnaugh l'expression $z = \overline{a\,c + b\,d}$

Solution Pour représenter d'une façon commode une expression sous forme d'une table de Karnaugh, il est préférable de mettre l'expression donnée sous forme d'une somme de produits. En effet, chaque produit est l'adresse d'une case ou d'un groupe de cases de la table de Karnaugh. D'où:

$$z = \overline{a\,c + b\,d}$$
$$= \overline{a\,c}\quad\overline{b\,d}$$
$$= (\bar{a} + \bar{c})\,(\bar{b} + \bar{d})$$

d'après les théorèmes de De Morgan.

Développons, on aura:

$$z = \bar{a}\,\bar{b} + \bar{a}\,\bar{d} + \bar{c}\,\bar{b} + \bar{c}\,\bar{d}$$

On a quatre variables, donc $n = 4$ et $p = q = n/2 = 2$. On aura donc une table de $2^2 = 4$ lignes et de $2^2 = 4$ colonnes.

Assignation des variables.

dc \ ba	00	01	11	10
00	1	1	1	1
01	1	0	0	1
11	1	0	0	0
10	1	1	0	0

Z

$\bar{a}\,\bar{b}$:	1re colonne, donc quatre 1 dans la première colonne		
$\bar{a}\,\bar{d}$:	$\bar{a}$:	1re et dernière colonnes	Inscrire 1 dans les cases
	$\bar{d}$:	1re et 2e lignes	de cette intersection
$\bar{c}\,\bar{b}$:	$\bar{b}$:	deux 1res colonnes	Inscrire 1 dans les cases
	$\bar{c}$:	1re et dernière lignes	de cette intersection

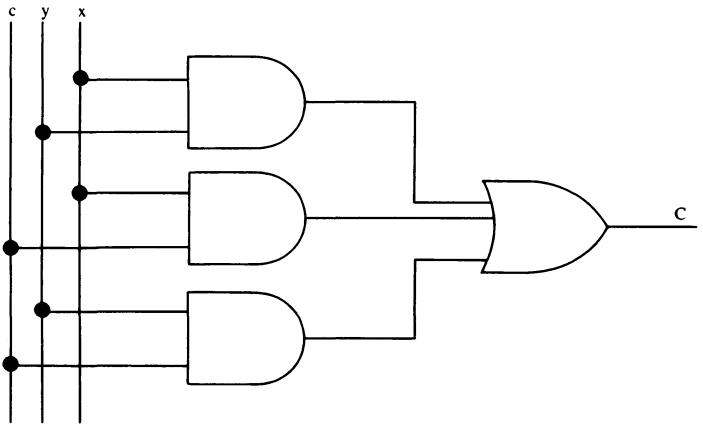
$\bar{c}\,\bar{d}$: 1re ligne, donc quatre 1 à la première ligne.

Compléter la table avec des 0.

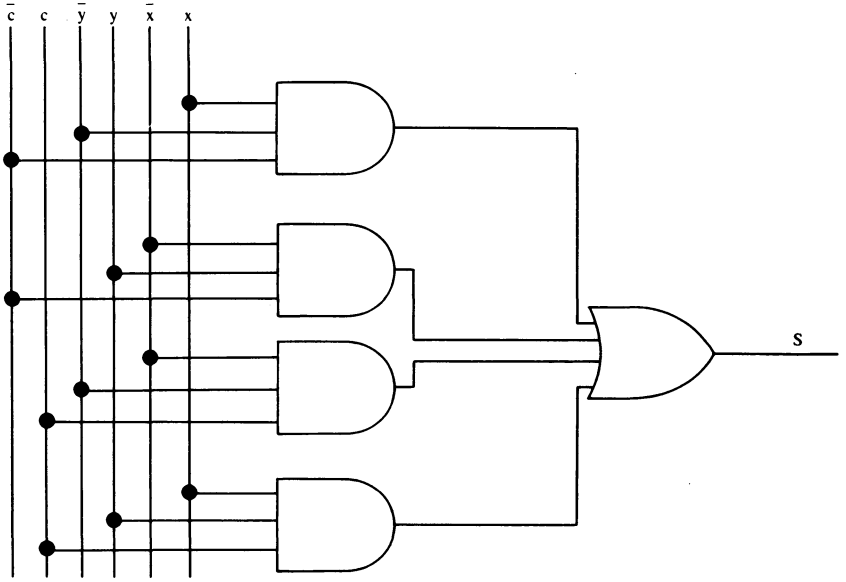
3-2-4 LOGIGRAMME, DIAGRAMME SYNOPTIQUE OU SCHÉMA LOGIQUE

Connaissant les symboles logiques, on peut construire immédiatement le logigramme de la fonction. On suppose qu'une série de lignes nous fournit les valeurs des variables d'entrées. On a alors les diagrammes logiques suivants:

a) Pour la fonction $C = xy + xc + yc$



b) Pour la fonction $S = x\bar{y}\bar{c} + \bar{x}y\bar{c} + \bar{x}\bar{y}c + xyc$



3-3 FORMES CANONIQUES

Les formes qui permettent de localiser chaque case d'une table de Karnaugh comportant un 1 ou un 0 ou chaque ligne d'une table de vérité comportant un 1 ou un 0 sont appelées *formes canoniques*. Ces formes sont en général simplifiables. Elles sont utiles dans certains cas de simplification ou de recherche d'implantations symétriques pour faciliter le dépannage.

On distingue quatre formes canoniques. La première est une somme de produits à implantation par des portes ET reliées à une porte OU. La deuxième est un produit de sommes à implantation par des portes OU reliées à une porte ET.

La troisième est la forme NON-ET et la quatrième, la forme NON-OU. Ces deux dernières formes sont implantées par un seul type de porte. Elles se déduisent des deux premières.

Nous donnerons la façon d'obtenir chacune de ces formes avant de donner des exemples.

3-3-1 PREMIÈRE FORME CANONIQUE: SOMME DE PRODUITS

Considérer la table de vérité ou la table de Karnaugh. À chaque 1 de la variable de sortie, faire correspondre un produit des n variables d'entrée sous la forme normale lorsque la variable d'entrée est à 1, sous la forme complément si la variable d'entrée est à 0. S'il y a p 1, faire la somme logique de ces p produits. Chaque produit doit contenir toutes les variables. L'expression obtenue est généralement simplifiable.

Dans notre cas, la table de vérité de la fonction S du paragraphe 3-2-2 permet d'écrire:

$$S = x\bar{y}\bar{c} + \bar{x}y\bar{c} + \bar{x}\bar{y}c + xyc$$

Lignes 2 3 5 8

On retrouve l'expression initiale de S.

Pour C on aura:

$$C = xy\bar{c} + x\bar{y}c + \bar{x}yc + xyc$$

On retrouve également l'expression initiale non simplifiée.

3-3-2 DEUXIÈME FORME CANONIQUE: PRODUIT DE SOMMES

Considérer la table de vérité ou la table de Karnaugh. À chaque 0 de la variable de sortie, faire correspondre une somme des n variables d'entrée sous la forme normale lorsque la variable d'entrée est égale à 0, sous la

forme complément si la variable d'entrée est égale à 1. S'il y a q 0, faire le produit logique de ces q sommes. Chaque somme doit contenir toutes les variables. L'expression obtenue est généralement simplifiable. Dans notre cas, la table de Karnaugh de la fonction S du paragraphe 3-2-3 permet d'écrire:

$$S = (x + y + c)(\bar{x} + \bar{y} + c)(\bar{x} + y + \bar{c})(x + \bar{y} + \bar{c})$$

Cases: 11 13 22 24

Cette forme peut se simplifier:

$$\begin{aligned} S &= (x\bar{x} + x\bar{y} + xc + y\bar{x} + y\bar{y} + yc + c\bar{x} + c\bar{y} + cc)(\bar{x}x + \bar{x}\bar{y} + \bar{x}\bar{c} \\ &\quad + yx + y\bar{y} + y\bar{c} + \bar{c}x + \bar{c}\bar{y} + \bar{c}\bar{c}) \\ &= (x\bar{y} + y\bar{x} + c)(\bar{x}\bar{y} + yx + \bar{c}) \\ &= x\bar{y}\bar{x}\bar{y} + x\bar{y}yx + x\bar{y}\bar{c} + y\bar{x}\bar{x}\bar{y} + y\bar{x}yx + y\bar{x}\bar{c} + c\bar{x}\bar{y} + \\ &\quad cyx + \bar{c}\bar{c} \\ &= x\bar{y}\bar{c} + \bar{x}y\bar{c} + \bar{x}\bar{y}c + xyc \end{aligned}$$

C'est bien l'expression de départ.

La première et la deuxième formes canoniques sont donc deux façons d'écrire la même fonction.

Deuxième forme canonique de C, à partir de sa table de Karnaugh du paragraphe 3-2-3:

$$C = (x + y + c)(\bar{x} + y + c)(x + \bar{y} + c)(x + y + \bar{c})$$

Cases: 11 12 14 21

Simplifions cette expression:

$$\begin{aligned} C &= (x + y + c)(\bar{x} + y + c)(x + \bar{y} + c)(x + y + \bar{c}) \\ &= (x\bar{x} + xy + xc + y\bar{x} + yy + yc + c\bar{x} + cy + cc)(xx + xy \\ &\quad + x\bar{c} + \bar{y}x + \bar{y}y + \bar{y}\bar{c} + cx + cy + c\bar{c}) \\ &= (y + c)(x + \bar{y}\bar{c} + cy) \\ &= yx + y\bar{y}\bar{c} + ycy + cx + c\bar{y}\bar{c} + ccy \\ &= xy + xc + yc \end{aligned}$$

qui est bien l'expression de départ de C.

3-3-3 TROISIÈME FORME CANONIQUE: FORME NON-ET

On la déduit de la première forme canonique, elle conduit à des diagrammes logiques n'utilisant que des portes NON-ET.

En effet, considérons une fonction logique qui se présente sous la forme d'une somme de produits, par exemple:

$$S = x\bar{y}\bar{c} + \bar{x}y\bar{c} + \bar{x}\bar{y}c + xyc$$

On a donc

$$S = \overline{\overline{x\bar{y}\bar{c} + \bar{x}y\bar{c} + \bar{x}\bar{y}c + xyc}}$$

puisqu'une double inversion est une opération d'effet nul.

D'après les théorèmes de De Morgan on obtient:

$$S = \overline{\overline{x\bar{y}\bar{c}} \quad \overline{\bar{x}y\bar{c}} \quad \overline{\bar{x}\bar{y}c} \quad \overline{xyc}}$$

On obtiendra de la même façon la troisième forme canonique de la fonction C, soit:

$$C = x\bar{y}\bar{c} + x\bar{y}c + \bar{x}yc + xyc$$

$$= \overline{\overline{x\bar{y}\bar{c} + x\bar{y}c + \bar{x}yc + xyc}}$$

$$= \overline{\overline{x\bar{y}\bar{c}} \quad \overline{x\bar{y}c} \quad \overline{\bar{x}yc} \quad \overline{xyc}}$$

3-3-4 QUATRIÈME FORME CANONIQUE: FORME NON-OU

Partir de la 2^e forme canonique, produit de sommes:

$$S = (x + y + c)(\bar{x} + \bar{y} + c)(\bar{x} + y + \bar{c})(x + \bar{y} + \bar{c})$$

On obtient la quatrième forme canonique de l'expression S:

$$S = \overline{\overline{x + y + c} + \overline{\bar{x} + \bar{y} + c} + \overline{\bar{x} + y + \bar{c}} + \overline{x + \bar{y} + \bar{c}}}$$

On obtiendra de la même façon la quatrième forme canonique de la fonction C, soit:

$$C = (x + y + c)(\bar{x} + y + c)(x + \bar{y} + c)(x + y + \bar{c})$$

$$= \overline{\overline{x + y + c} + \overline{\bar{x} + y + c} + \overline{x + \bar{y} + c} + \overline{x + y + \bar{c}}}$$

Exemple 1 Écrire les quatre formes canoniques de la fonction donnée par la table de vérité ci-dessous:

z	y	x	t
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Solution

Première forme canonique, somme de produits

$$t = x\bar{y}\bar{z} + \bar{x}y\bar{z} + \bar{x}\bar{y}z + x\bar{y}z$$

Deuxième forme canonique, produit de sommes

$$t = (x + y + z)(\bar{x} + \bar{y} + z)(x + \bar{y} + \bar{z})(\bar{x} + \bar{y} + \bar{z})$$

Troisième forme canonique, forme NON-ET déduite de la première forme:

$$t = \overline{x\bar{y}\bar{z} + \bar{x}y\bar{z} + \bar{x}\bar{y}z + x\bar{y}z}$$

$$= \overline{x\bar{y}\bar{z}} \quad \overline{\bar{x}y\bar{z}} \quad \overline{\bar{x}\bar{y}z} \quad \overline{x\bar{y}z}$$

Quatrième forme canonique, forme NON-OU déduite de la deuxième forme:

$$t = \overline{(x + y + z)(\bar{x} + \bar{y} + z)(x + \bar{y} + \bar{z})(\bar{x} + \bar{y} + \bar{z})}$$

$$= \overline{x + y + z} + \overline{\bar{x} + \bar{y} + z} + \overline{x + \bar{y} + \bar{z}} + \overline{\bar{x} + \bar{y} + \bar{z}}$$

Exemple 2 Écrire les quatre formes canoniques de l'expression

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

Solution

Première méthode:

Construisons la table de vérité. On obtient:

c	b	a	z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Première forme canonique:

$$z = a\,b\,\bar{c} + \bar{a}\,\bar{b}\,c + a\,b\,c$$

Deuxième forme canonique:

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

Troisième forme canonique:

$$z = \overline{\overline{a\,b\,\bar{c}}\quad \overline{\bar{a}\,\bar{b}\,c}\quad \overline{a\,b\,c}}$$

Quatrième forme canonique:

$$z = \overline{\overline{a+b+c} + \overline{\bar{a}+b+c} + \overline{a+\bar{b}+c} + \overline{\bar{a}+b+\bar{c}} + \overline{a+\bar{b}+\bar{c}}}$$

Deuxième méthode:

On remarque que chaque terme d'une forme canonique contient toutes les variables sous la forme normale ou complémentée. Dans l'expression donnée, un des termes n'a que deux variables. Pour obtenir la forme canonique il faut lui ajouter la troisième variable sous forme normale et complémentée, on a alors:

$$z = a\,b\,(c + \bar{c}) + \bar{a}\,\bar{b}\,c$$

$$= a\,b\,c + a\,b\,\bar{c} + \bar{a}\,\bar{b}\,c$$: première forme canonique, d'où

$$z = \overline{\overline{a\,b\,c}\quad \overline{a\,b\,\bar{c}}\quad \overline{\bar{a}\,\bar{b}\,c}}$$: troisième forme canonique

REMARQUE Toutes les combinaisons de cba prises dans cet ordre (a en position du bit de poids faible) sont des nombres binaires de 0 à 7. (Voir la table de vérité ci-dessus). On peut donc écrire l'expression z sous forme numérique en octal par exemple comme: z_8 (c,b,a) = R(3,4,7) où 3 représente c b a, 4 représente c $\bar{b}$ $\bar{a}$ et 7 représente cba. Une expression mise sous cette forme est dite *numérique*. La lettre R, mise pour *réunion*, rappelle que l'on a une somme de produits dont la première forme canonique. La troisième forme canonique s'en déduit aisément.

La deuxième forme se déduit de la première en prenant tous les nombres de 0 à 7 qui ne sont pas inclus dans la première forme.

On obtient:

$z_8 = I(0,1,2,5,6)$ I est mis pour intersection, d'où:

$$z_8 = \underset{0}{(c+b+a)}\ \underset{1}{(c+b+\bar{a})}\ \underset{2}{(c+\bar{b}+a)}\ \underset{5}{(\bar{c}+b+\bar{a})}\ \underset{6}{(\bar{c}+\bar{b}+a)}$$

On peut en déduire facilement la quatrième forme canonique.

Exemple 3 Écrire les quatre formes canoniques de l'expression Z suivante donnée par sa table de Karnaugh.

c \ ba	00	01	11	10	
0	1	1	0	1	Z
1	1	0	0	0	

Solution

Première forme: $Z = \bar{a}\bar{b}\bar{c} + a\bar{b}\bar{c} + \bar{a}b\bar{c} + \bar{a}\bar{b}c$

Deuxième forme: $Z = (\bar{a}+\bar{b}+c)(\bar{a}+b+\bar{c})(a+\bar{b}+\bar{c})(\bar{a}+\bar{b}+\bar{c})$

Troisième forme: $Z = \overline{\bar{a}\bar{b}\bar{c}\ \ a\bar{b}\bar{c}\ \ \bar{a}b\bar{c}\ \ \bar{a}\bar{b}c}$

Quatrième forme: $Z = \overline{\bar{a}+\bar{b}+c} + \overline{a+\bar{b}+c} + \overline{\bar{a}+b+\bar{c}} + \overline{a+b+c}$

REMARQUE Sous forme simplifiée on a: $Z = \bar{b}\bar{c} + \bar{a}\bar{c} + \bar{a}\bar{b}$. Elle peut être déduite algébriquement de la première forme canonique ou par une méthode qu'on verra plus loin.

Une autre forme sera: $Z = (\bar{a} + \bar{b})(\bar{a} + \bar{c})(\bar{b} + \bar{c})$ déduite de la deuxième forme canonique.

Ou encore: $Z = \overline{\overline{b}\,\overline{c}} \quad \overline{\overline{a}\,\overline{c}} \quad \overline{\overline{a}\,\overline{b}}$ et $Z = \overline{\overline{a}+\overline{b}} + \overline{\overline{a}+\overline{c}} + \overline{\overline{b}+\overline{c}}$

Les formes canoniques sont les expressions booléennes les plus complètes. Chacune de ces formes est simplifiable.

3-4 SIMPLIFICATION PAR LA TABLE DE KARNAUGH

3-4-1 INTRODUCTION

L'algèbre booléenne, comme on le verra dans le prochain chapitre, sert à résoudre des problèmes techniques. Ces problèmes techniques sont schématisés sous forme de tables de vérité ou de tables de Karnaugh. L'objectif est d'obtenir une ou plusieurs sorties bien définies répondant à certains conditions bien déterminées à l'entrée.

De la table de vérité, ou de la table de Karnaugh, on déduit une expression algébrique. Comme on l'a vu, cette expression algébrique donne un logigramme qui est le schéma de principe de l'implantation de la fonction. Cette expression algébrique a différentes formes suivant les contraintes techniques. Parmi les formes on a déjà vu la forme somme de produits pour des implantations OU-ET ou la forme produit de sommes pour des implantations ET-OU. On a vu aussi la forme NON-OU et la forme NON-ET. La forme d'écriture dépend donc du type des modules logiques qu'on veut utiliser.

On peut être aussi contraint de choisir des formes symétriques pour des raisons de facilité de dépannage ou des formes minimales qui ne sont pas les plus simplifiées dans des cas précis en logique séquentielle, par exemple, pour éviter les aléas de fonctionnement.

La méthode que nous retiendrons minimise le nombre de variables. Ces variables, ou lettres, seront prises sous forme normale ou inverse, ces deux formes étant, en général, disponibles dans les circuits. On compte les lettres chaque fois qu'elles apparaissent avec ou sans barre.

La fonction précédente $Z = \overline{a}\,\overline{b}\,\overline{c} + a\,\overline{b}\,\overline{c} + \overline{a}\,b\,\overline{c} + \overline{a}\,\overline{b}\,c$ comporte douze lettres. Pour l'implanter il faut quatre portes ET à trois entrées et une porte OU à quatre entrées.

Sa forme simplifiée $Z = \overline{b}\,\overline{c} + \overline{a}\,\overline{c} + \overline{a}\,\overline{b}$ comprend six lettres, son implantation nécessite trois portes ET à deux entrées et une porte OU à trois entrées.

La forme $Z = \overline{b}\,\overline{c} + \overline{a}\,(\overline{c} + \overline{b})$ comprend cinq lettres, son implantation requiert deux portes ET à deux entrées et deux portes OU à deux entrées soit encore quatre portes au total.

La forme $Z = \overline{\overline{b}\,\overline{c} \quad \overline{a}\,\overline{c} \quad \overline{a}\,\overline{b}}$ nécessite trois portes NON-ET à deux entrées et une porte NON-ET à trois entrées. Soit quatre portes au total.

On peut aussi chercher à diminuer le nombre de circuits intégrés. Chacune des trois dernières formes nécessite deux circuits intégrés. En guise d'exercice, trouver lesquels par consultation d'un catalogue de fabricant.

3-4-2 SIMPLIFICATION À L'AIDE D'UNE TABLE COMPLÈTE

Nous avons vu que les lignes et les colonnes des tables de Karnaugh sont numérotées selon le code Gray ou binaire réfléchi. On sait que, dans ce code, deux nombres adjacents, c'est-à-dire l'un à la suite de l'autre, ne sont différents que par un seul bit. On se servira de cette propriété d'adjacence du code Gray pour les simplifications. Prenons des exemples:

tz \ yx	00	01	11	10
00	1	0	0	0
01	1	0	0	0
11	0	0	0	0
10	0	0	1	1

A

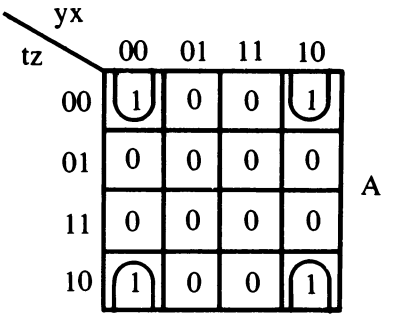
Selon la première forme canonique, on a $A = \overline{x}\,\overline{y}\,\overline{z}\,\overline{t} + \overline{x}\,\overline{y}\,z\,\overline{t} + x\,\overline{y}\,\overline{z}\,\overline{t} + x\,\overline{y}\,z\,\overline{t}$. Les deux premiers termes correspondent aux deux 1 en haut à gauche. Si on met $\overline{x}\,\overline{y}\,\overline{t}$ en facteur dans ces termes on aura: $\overline{x}\,\overline{y}\,\overline{t}\,(z + \overline{z}) = \overline{x}\,\overline{y}\,\overline{t}.\ (1) = \overline{x}\,\overline{y}\,\overline{t}$. Les deux termes sont dépendants de $\overline{z}$ et z; leur somme est indépendante de $\overline{z}$ et de z.

De la même façon pour les deux termes correspondants aux 1 en bas à droite, on obtient: $y\,\overline{z}\,t\,(x + \overline{x}) = y\,\overline{z}\,t$.

D'où l'expression simplifiée: $A = \overline{x}\,\overline{y}\,\overline{t} + y\,\overline{z}\,t$, au lieu de seize lettres on n'en a que six. Ce qui nous permet d'énoncer cette première règle comme il suit.

PREMIÈRE RÈGLE On peut regrouper deux 1 adjacents dans une somme de produits en éliminant la variable qui change d'état.

Il faut noter que l'adjacence existe aussi pour les extrémités de la table. Soit l'exemple ci-dessous:



Si on prend les deux 1 de la première colonne, on obtient $\bar{x}\,\bar{y}\,\bar{z}$ car ces 1 dépendant de t ou de $\bar{t}$ leur regroupement est donc indépendant de t et de t. Si on prend les deux 1 de la dernière colonne on obtient l'expression $x\,\bar{y}\,\bar{z}$.

Si on prend les deux 1 de la première ligne, on a l'expression $\bar{x}\,\bar{z}\,t$.

Si on prend les deux 1 de la dernière ligne, on a l'expression $x\,z\,t$.

Si on peut regrouper les 1 par quatre, on obtient une expression encore plus simple. Il faut quatre 1 adjacents, c'est-à-dire dans un carré, une ligne, une colonne ou aux extrémités de la table.

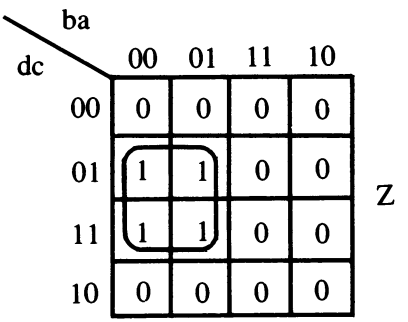
Dans l'exemple précédent on a obtenu les expressions:

$\bar{x}\,\bar{y}\,\bar{z}$ pour la première colonne et

$\bar{x}\,y\,\bar{z}$ pour la dernière.

La réunion de ces deux expressions regroupe les quatre 1 donc $A = \bar{x}\,\bar{y}\,\bar{z} + \bar{x}\,y\,\bar{z} = \bar{x}\,\bar{z}\,(y + \bar{y})$. Ces deux expressions dépendent de y et $\bar{y}$, leur somme est donc indépendante de y et de $\bar{y}$ et s'écrit $A = \bar{x}\,\bar{z}$, en deux lettres.

De même pour le tableau ci-dessous



on a: $Z = \bar{b}c$

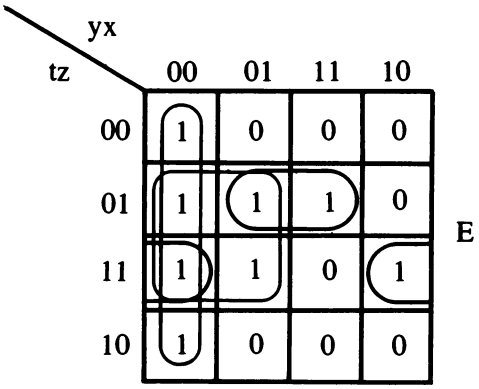
Ce qui nous permet d'énoncer cette deuxième règle comme il suit:

DEUXIÈME RÈGLE On peut regrouper quatre 1 adjacents dans une somme de produits et éliminer les deux variables qui changent d'état.

Cette règle se généralise à toutes les puissances de 2: 2, 4, 8 etc.

Il est donc intéressant de former des groupes de 1 aussi importants que possible: 2, 4, 8, . . .

Soit l'expression E représentée par la table de Karnaugh ci-dessous:



On peut regrouper les quatre 1 de la première colonne, ce qui donne $\bar{x}\,\bar{y}$.

On peut regrouper les quatre 1 des deuxième et troisième lignes, des première et deuxième colonnes, on obtient $\bar{y}\,z$. Ce qui nous donne jusqu'à présent $\bar{x}\,\bar{y} + \bar{y}\,z$ (1).

On a pris deux fois les 1 à l'intersection des 2^e et 3^e lignes avec la 1^{ere} colonne, mais selon la loi A + A = A, cela n'entraîne aucune erreur. Un même 1 peut être introduit dans plusieurs groupes. En effet si on ne prend que les deux 1 de droite, deuxième colonne, on aura $x\,\bar{y}\,z$ dont la somme avec la première colonne donnera:

$$(2)\ \bar{x}\,\bar{y} + x\,\bar{y}\,z = \bar{y}\,(\bar{x} + x\,z)$$
$$= \bar{y}\,(\bar{x} + z)$$
$$= \bar{x}\,\bar{y} + \bar{y}\,z$$

On voit que les sommes partielles (1) et (2) sont bien égales.

Les deux 1 de la deuxième ligne, des deuxième et troisième colonnes donnent $x\,z\,\bar{t}$ et les deux 1 extrêmes de la troisième ligne donnent $\bar{x}\,z\,t$.

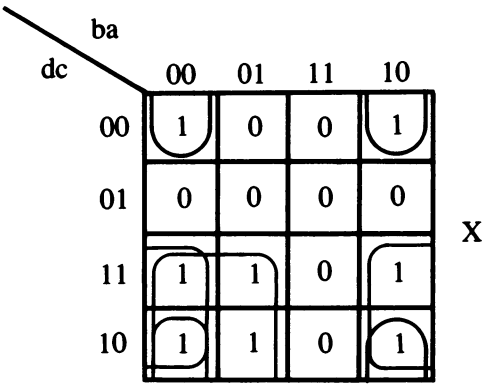
D'où: $E = \bar{x}\,\bar{y} + \bar{y}\,z + x\,z\,\bar{t} + \bar{x}\,z\,t$

qui est une forme simplifiée de l'expression algébrique de la table de Karnaugh.

Ce qui nous permet d'énoncer cette troisième règle comme il suit.

TROISIÈME RÈGLE Un même 1 peut être introduit dans plusieurs groupes.

Exemple 1 Chercher l'expression qui donne une implantation, avec le minimum de portes NON-ET, de la fonction X donnée par la table de Karnaugh ci-dessous:



Solution

On a: $X = \bar{b}\,d + \bar{a}\,d + \bar{a}\,\bar{c}$

Mis sous forme NON-ET, $X = \overline{\bar{b}\,d}\ \overline{\bar{a}\,d}\ \overline{\bar{a}\,\bar{c}}$, soit trois portes NON-ET à deux entrées et une porte NON-ET à trois entrées.

REMARQUE La forme produit de sommes se prête aux même règles que la forme somme de produits.

Prenons les 0 de la table de Karnaugh ci-dessus, par exemple, alors:

— la troisième colonne donne $(\bar{a} + \bar{b})$

— la deuxième ligne donne $(\bar{c} + d)$

— le carré du haut (deuxième et troisième colonnes) donne $(\bar{a} + d)$

D'où: $X = (\bar{a} + \bar{b})\,(\bar{c} + d)\,(\bar{a} + d)$

$= (\bar{a}\,\bar{c} + \bar{a}\,d + \bar{b}\,\bar{c} + \bar{b}\,d)\,(\bar{a} + d)$

$\bar{a}\,\bar{c}\,\bar{a} + \bar{a}\,\bar{c}\,d + \bar{a}\,d\,\bar{a} + \bar{a}\,d\,d + \bar{b}\,\bar{c}\,\bar{a} + \bar{b}\,\bar{c}\,d + \bar{b}\,d\,\bar{a} + \bar{b}\,d\,d$

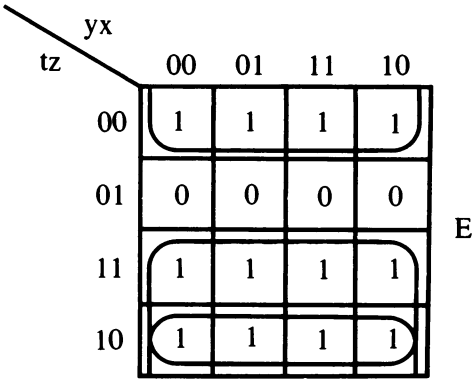
$= \bar{b}\,d + \bar{a}\,d + \bar{a}\,\bar{c}$

C'est bien l'expression que l'on a trouvé plus haut.

Expression NON-OU: $X = \overline{\overline{\bar{a} + \bar{b}} + \overline{\bar{c} + d} + \overline{\bar{a} + d}}$, soit trois portes NON-OU à deux entrées et une porte NON-OU à trois entrées.

La table de Karnaugh est un outil de simplification très commode, beaucoup plus puissant que la méthode algébrique. Elle permet d'écrire très rapidement l'expression minimale de l'implantation désirée.

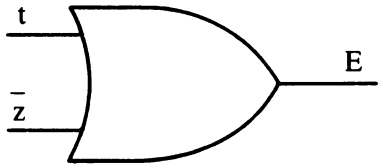
Exemple 2 Implanter avec le minimum de portes la fonction E donnée par la table de Karnaugh ci-dessous:



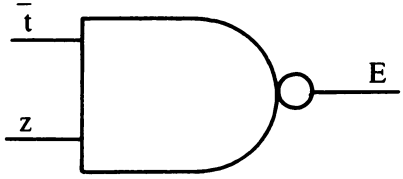
Solution

$E = t + \bar{z}$ pour les 1

$E = t + \bar{z}$

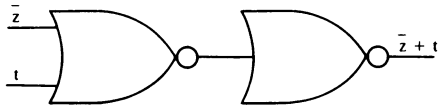


$E = \overline{\bar{t}z}$



$E = \bar{z} + t$ pour les 0: même expression et mêmes implantations que ci-dessus.

$E = \overline{\overline{\bar{z} + t}}$

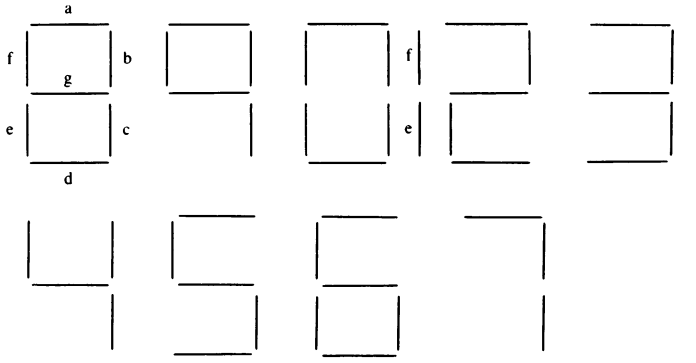


Les implantations comprenant soit une porte OU, soit une porte NON-ET sont minimales.

3-4-3 SIMPLIFICATION À L'AIDE D'UNE TABLE INCOMPLÈTE

Dans les problèmes techniques il arrive que la fonction logique ne soit pas spécifiée pour certaines combinaisons des variables d'entrée. Cela survient lorsque les combinaisons ne se présentent jamais en fonctionnement normal.

Exemple Décoder un nombre BCD pour commander un *segment* d'un indicateur à sept segments notés a, b, c, d, e, f, g constituant les chiffres décimaux comme il apparaît ci-dessous:



Solution Considérons, par exemple, le segment b. Il entre dans la constitution de 8, 9, 0, 2, 3, 4, 7.

Si on inscrit 1 dans les cases de la table de Karnaugh où b est allumé, (8, 9, 0, 2, 3, 4, 7) et 0 dans celles où il est éteint (1, 5, 6) on obtient la table ci-dessous.

Pour un nombre BCD désigné par DCBA, A est le bit de poids faible.

DC \ BA	00	01	11	10
00	1	0	1	1
01	1	0	1	0
11	X	X	X	X
10	1	1	X	X

b

Les dix cases du code BCD sont remplies mais les cases correspondant à 10, 11, 12, 13, 14, 15 ne sont pas définies puisque le code BCD n'est défini que pour les nombres décimaux de 0 à 9.

On dit que les valeurs dans ces cases sont *indifférentes* et on les désigne par X.

Ces valeurs indifférentes seront choisies de manière à faciliter la formation des groupes de 1. On peut attribuer n'importe quelle valeur à ces cases puisque les combinaisons correspondantes ne se présentent jamais en temps normal, d'où:

$$b = \bar{A}\bar{B} + AB + \bar{C}B + D$$

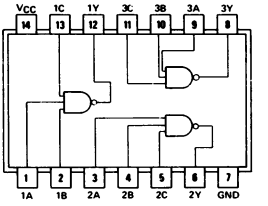
Nous verrons d'autres applications lorsque nous étudierons le décodage.

3-5 QUELQUES CIRCUITS INTÉGRÉS D'IMPLANTATION D'UNE FONCTION LOGIQUE

Nous avons déjà signalé les circuits intégrés implantant des fonctions simples au chapitre précédent. Nous allons en signaler d'autres très courants aussi.

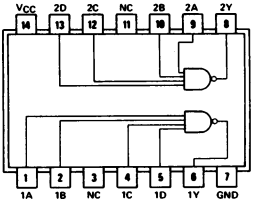
Circuit intégré 7410: trois portes NON-ET à trois entrées
Circuit intégré 7420: deux portes NON-ET à quatre entrées
Circuit intégré 7430: une porte NON-ET à huit entrées

7410



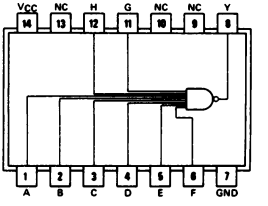
$Y = \overline{ABC}$

7420



$Y = \overline{ABCD}$

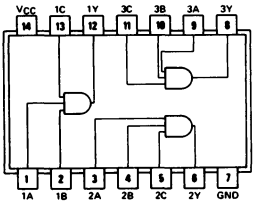
7430



$Y = \overline{ABCDEFGH}$

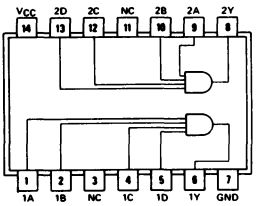
Circuit intégré 7411: trois portes ET à trois entrées
Circuit intégré 7421: deux portes ET à quatre entrées

7411



Y = ABC

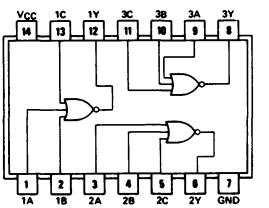
7421



Y = ABCD

Circuit intégré 7427: trois portes NON-OU à trois entrées

7427

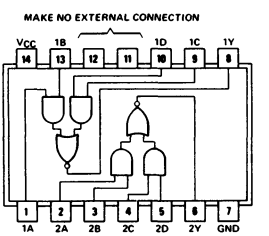


$Y = \overline{A+B+C}$

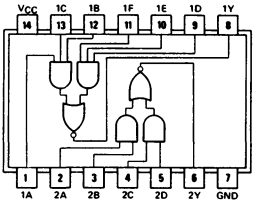
Il existe un grand nombre de circuits intégrés à fonction ET, OU, INVERSEUR. Ils sont désignés par le sigle anglais AOI (*And Or Invert*).

On peut signaler le circuit intégré 7451 qui existe en deux versions comme le montre le schéma ci-dessous:

7451



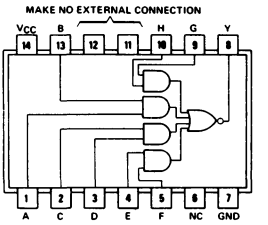
$Y = \overline{AB+CD}$



$1Y = \overline{(1A \cdot 1B \cdot 1C)+(1D \cdot 1E \cdot 1F)}$
$2Y = \overline{(2A \cdot 2B)+(2C \cdot 2D)}$

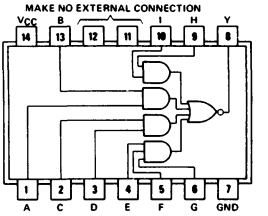
Signalons également un autre circuit intégré AOI, le 7454 dont différentes versions apparaissent ci-dessous.

7454



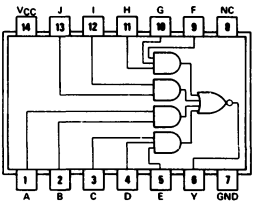
$Y = \overline{AB+CD+EF+GH}$

SN5454 (J) **SN7454 (J, N)**



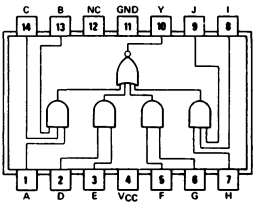
$Y = \overline{AB+CD+EFG+HI}$

SN54H54 (J) **SN74H54 (J, N)**



$Y = \overline{AB+CDE+FGH+IJ}$

SN54L54 (J) **SN74L54 (J, N)**
SN54LS54 (J, W) **SN74LS54 (J, N)**



$Y = \overline{ABC+DE+FG+HIJ}$

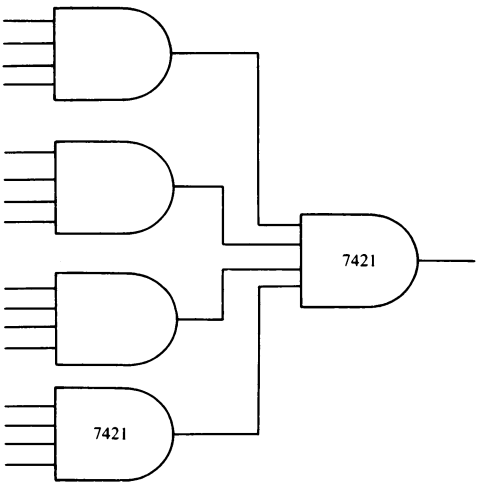
SN54L54 (T)

3-6 IMPLANTATION D'UNE FONCTION LOGIQUE À GRAND NOMBRE DE VARIABLES

Une bonne connaissance de la manipulation des fonctions logiques permet d'implanter des fonctions comprenant un très grand nombre de variables. Nous allons signaler quelques solutions pour des fonctions élémentaires.

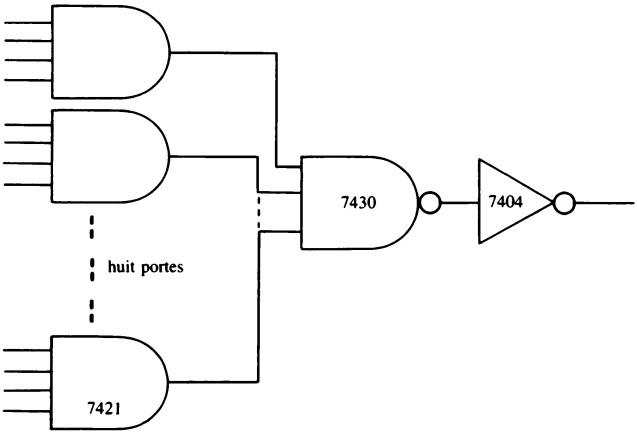
3-6-1 FONCTION ET

Il existe des circuits intégrés ET à deux, trois et quatre entrées. Pour un nombre d'entrées plus élevé on peut, par exemple, recourir à l'implantation ci-dessous à l'aide de CI 7421 qui permet seize entrées pour deux niveaux de circuits intégrés.



Implantation ET à seize entrées

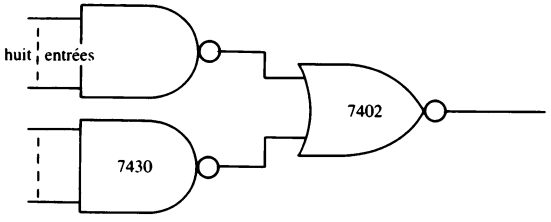
Pour trois niveaux de circuits on peut avoir soixante-quatre entrées (quatre fois l'implantation précédente et une porte ET à quatre entrées), ou bien on peut utiliser comme ci-dessous une porte NON-ET à huit entrées suivie d'un inverseur. Dans ce cas, on aura une implantation ET à trente-deux entrées.



Implantation ET à trente-deux entrées

La porte NON-OU permet aussi une implantation ET avec une inversion des entrées. Par exemple, une implantation ET à seize entrées peut être réalisée par la disposition suivante, selon la relation

$$\overline{\overline{ABC \ldots} + \overline{XYZ \ldots}} = ABC \ldots XYZ \ldots$$



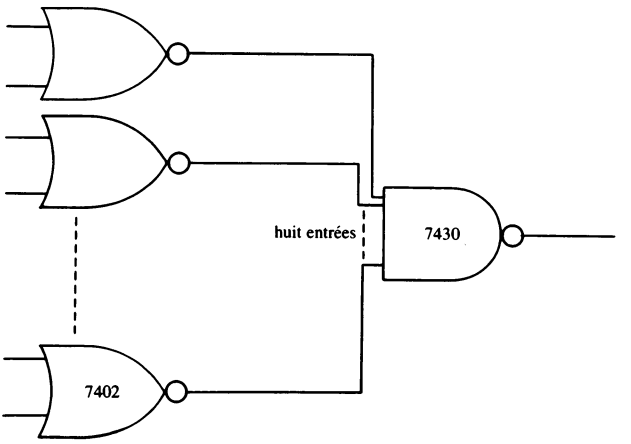
Implantation ET à seize entrées

3-6-2 FONCTION OU

Les fonctions OU ne sont pas nombreuses dans la série 74XX. Le circuit intégré OU 7432 est le plus courant. On peut utiliser des portes NON-OU avec inversion de la sortie. La combinaison NON-OU, NON-ET ci-

dessous permet une implantation OU à seize entrées, d'après la relation

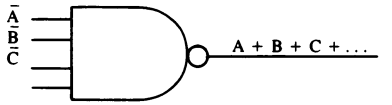
$$\overline{\overline{(A + B)} \cdot \overline{(C + D)} \cdot \overline{(E + F)} \ldots} = A + B + C + D + E + F + \ldots$$



Implantation OU à seize entrées

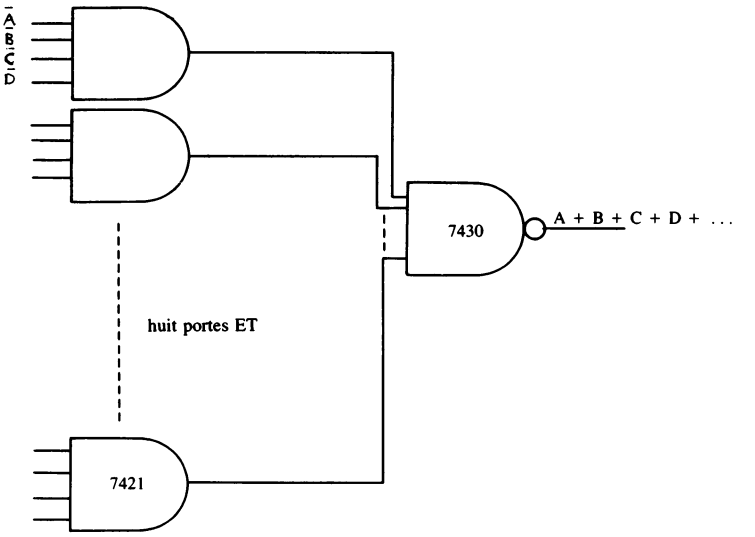
Trois implantations de ce genre reliées à une porte NON-OU 7427 à trois entrées donnera une implantation NON-OU à quarante-huit entrées.

Une porte NON-ET avec inversion des entrées donne un circuit OU selon la relation $\overline{\overline{A} \cdot \overline{B} \cdot \overline{C} \ldots} = A + B + C + \ldots$ déduite d'un théorème de De Morgan, soit:



Implantation OU à l'aide d'une porte NON-ET

On peut, comme dans le cas de la fonction ET, utiliser des portes ET avant d'entrer dans la porte NON-ET. L'implantation OU ci-dessous permet jusqu'à trente-deux entrées.

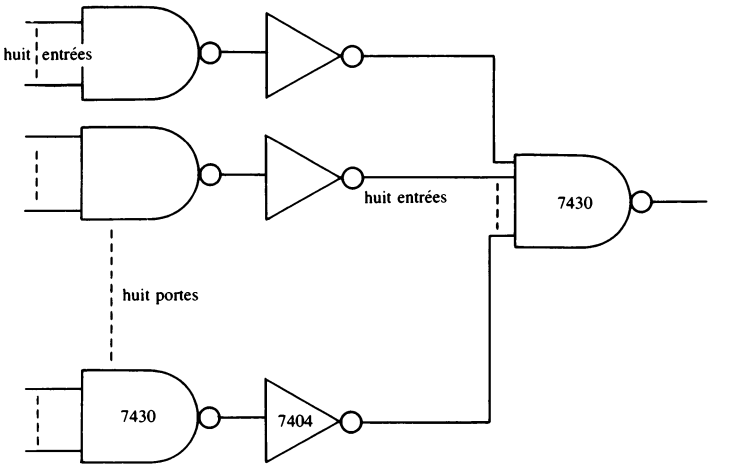


Implantation OU à trente-deux entrées

3-6-3 FONCTION NON-ET

Il existe, en circuit intégré, une porte NON-ET permettant jusqu'à huit entrées (deux, trois, quatre, huit entrées).

Pour augmenter le nombre d'entrées, on peut utiliser des portes NON-ET ou des portes ET.

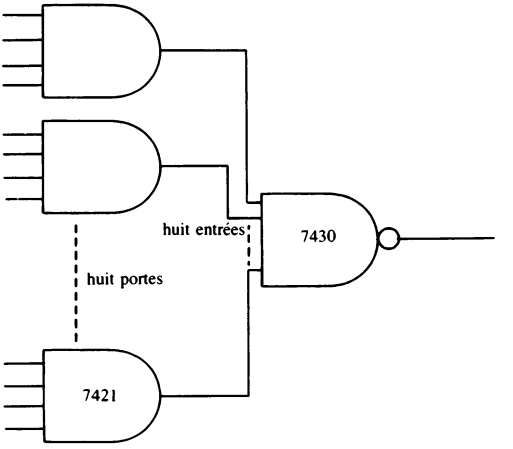


Implantation NON-ET à soixante-quatre entrées

Pour le schéma avec des portes NON-ET on utilise la propriété

$$\overline{A \cdot B \cdot C \cdot \ldots} \cdot \overline{A_1B_1C_1 \ldots} \cdot \overline{A_2B_2C_2 \ldots}$$
$$= \overline{ABC \ldots \quad \cdot A_1B_1C_1 \ldots \cdot A_2B_2C_2 \ldots}$$

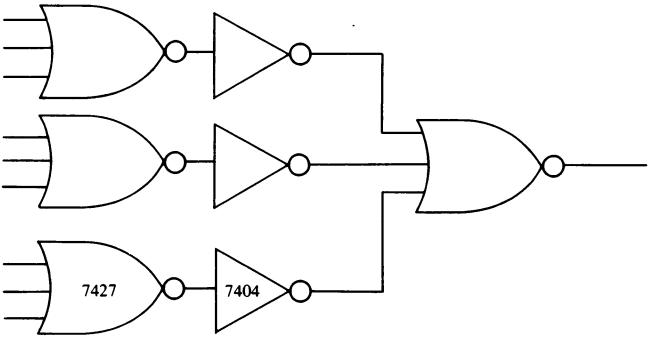
Le montage avec des portes ET permet jusqu'à trente-deux entrées.



Implantation NON-ET à trente-deux entrées

3-6-4 FONCTION NON-OU

Avec les portes NON-OU à trois entrées on peut obtenir une implantation NON-OU à neuf entrées en ajoutant des inverseurs.



Implantation NON-OU à neuf entrées

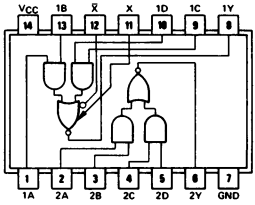
Si on veut six entrées ont peut prendre trois circuits intégrés OU 7432 entrant directement dans une porte NON-OU à trois entrées.

3-6-5 CIRCUITS INTÉGRÉS AOI

Certains circuits AOI sont expansibles. C'est le cas par exemple du circuit 7450 dont le schéma des broches est donné ci-dessous. On peut lui ajouter au maximum quatre circuits 7460 pour former, par exemple:

$Y = \overline{AB + CD + A_1B_1C_1D_1}$ avec un circuit supplémentaire.

7450



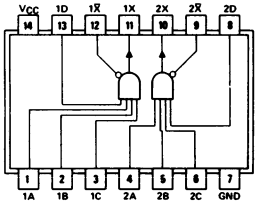
$Y = \overline{AB+CD+X}$

SN5450 (J) SN7450 (J, N)
SN54H50 (J) SN74H50 (J, N)

Opérateur NON − OU − ET + expansion

X et $\overline{X}$: sorties du 7460

7460



SN5460 (J) SN7460 (J, N)
SN54H60 (J) SN74H60 (J, N)

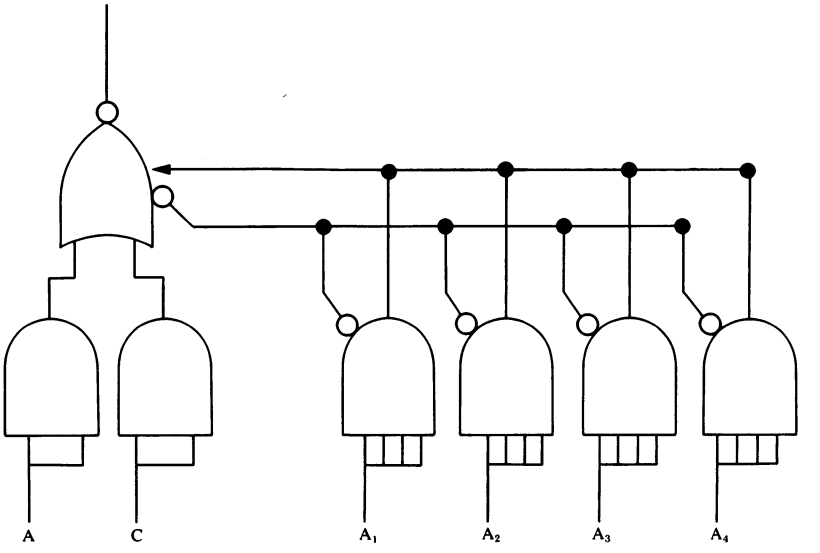
Expanseur à quatre entrées
X = ABCD lorsque branché aux entrées X et $\overline{X}$ du 7450.

et $Y = \overline{AB + CD + A_1B_1C_1D_1 + A_2B_2C_2D_2 + A_3B_3C_3D_3 + A_4B_4C_4D_4}$ en utilisant le maximum permis.

Ces circuits sont aussi utiles dans l'implantation de fonctions simples.

Si, par exemple, on met sur chaque groupe d'entrées de la dernière fonction Y la même variable, (toutes les entrées d'un groupe reliées à un même point d'entrée), on obtient: $Y = \overline{A + C + A_1 + A_2 + A_3 + A_4}$ c'est une implantation NON-OU à six entrées. Si on inverse cette sortie, on a une implantation OU à six entrées. Si on prend les inverses des variables, on a à la sortie une implantation ET à six entrées.

$$Y = \overline{\bar{A} + \bar{C} + \bar{A}_1 + \bar{A}_2 + A_3 + \bar{A}_4}$$
$$= A \cdot C \cdot A_1 \cdot A_2 \cdot A_3 \cdot A_4$$



Implantation NON-OU à six entrées à l'aide de 7450 et 7460

PROBLÈMES

Représenter les expressions E des problèmes 3-1 à 3-4 sous forme de:

a) table de vérité dans l'ordre binaire naturel
b) table de Karnaugh (en déduire, si possible, une forme simplifiée)
c) logigramme

3-1 $E = a\,b + a\,\bar{c}$

3-2 $E = \bar{a}\,b\,c + \bar{a}\,\bar{b}\,c + a\,\bar{b}\,c$

3-3 $E = a\,\overline{b\,c} + \bar{a}\,\overline{b\,c} + \bar{a}\,b\,\bar{c} + \overline{a\,b}\,c$

3-4 $E = \bar{a}\,b\,c + a\,\bar{b}\,c + a\,b\,\bar{c}$

Représenter les expressions E des problèmes 3-5 à 3-12

a) sous la forme d'une table de Karnaugh
b) sous la première forme canonique
c) sous la deuxième forme canonique
d) sous forme simplifiée
e) sous forme d'un logigramme en n'utilisant que des portes NON-ET
f) sous forme d'un logigramme en n'utilisant que des portes NON-OU.
(des bascules fournissent les variables sous forme normale et inversée).

3-5 $E = (a + b)\,(\bar{a} + \bar{b} + \bar{c})\,(a + c)$

3-6 $E = (a + d)\,(a\,b + a\,c)\,(\bar{a}\,\bar{c} + b)$

3-7 $E = (a + c + d)\,(b + c + d)$

3-8 $E = (a\,\bar{b} + c + c\,d)\,(\bar{a}\,\bar{c} + b\,c + d)$

3-9 $E = (a\,\bar{b} + a\,b + a\,c)\,(\bar{a}\,\bar{b} + a\,b + a\,\bar{c})$

3-10 $E = a\,b\,c + a\,\overline{b\,c} + \bar{a}\,\overline{b\,c}$

3-11 $E = \bar{a}\,b\,c + a\,\bar{b}\,c + a\,b\,\bar{c}$

3-12 $E = a\,b\,c + \bar{a}\,b\,c + a\,\bar{b}\,c + a\,b\,\bar{c} + a\,\overline{b\,c} + \bar{a}\,b\,\bar{c} + \bar{a}\,\bar{b}\,\bar{c}$

3-13 Soit l'expression $y = \bar{a}\,c\,\bar{d} + a\,b\,\bar{c}$

On suppose que toutes les variables sont disponibles sous les deux formes (normale et complémentée).

a) fair le diagramme de câblage de cette expression en supposant que vous n'avez que des portes ET et OU à deux entrées (minimiser le nombre de portes).

b) même question si vous avez seulement des portes NON-ET standard TTL (à deux, trois, quatre ou huit entrées).

3-14 Soit la table de Karnaugh ci-dessous:

dc \ ba	00	01	11	10
00	1	0	0	0
01	0	0	0	0
11	1	0	0	1
10	1	0	0	0

E

a) écrire la première forme canonique de l'expression E

b) écrire la forme algébrique simplifiée de cette expression

c) faire le câblage (logigramme) avec seulement des portes NON-OU.

Soit les expressions y des problèmes 3-15 et 3-16.

a) les représenter sous forme de table de Karnaugh

b) déterminer la forme simplifiée de $\bar{y}$.

3-15 $y = \bar{a}\,\bar{b}\,c\,d + a\,c\,d + \bar{a}\,\bar{c}\,d$

3-16 $y = \bar{c}\,\bar{d} + \bar{a}\,c\,d + a\,\bar{c}\,d$

3-17 Écrire sous forme d'un produit de sommes l'expression

$Y = AB\bar{C} + DE + \bar{A}E$

4

Problèmes de logique combinatoire

4-1 OBJECTIFS

1. Savoir résoudre des problèmes de logique combinatoire.
2. Savoir implanter, à l'aide de circuits intégrés, leur fonction solution.

Définissons tout d'abord les variables d'entrée et de sortie. Aux entrées sont assignées des lettres appelées *variables d'entrée*. De la même façon, les lettres assignées aux sorties seront appelées *variables de sortie*. Dans tout problème de logique combinatoire, les variables de sortie dépendent des variables d'entrée et d'elles seules.

PROBLÈME 1 — DEMI-ADDITIONNEUR

Concevoir un circuit capable d'additionner deux bits, capable donc de générer leur somme $\sum$ et leur report C*.

Solution

Étape 1 — Table de vérité

Appelons A et B les deux variables d'entrée représentant les bits à additionner. On a, par définition de l'addition binaire, la table de vérité suivante:

Entrées		Sorties	
B	A	$\sum$	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

On définit donc la valeur binaire des sorties, pour toutes les combinaisons possibles des entrées. Pour q variables d'entrée la table de vérité aura 2^q lignes qui correspondront aux nombres binaires de 0 à $2^q - 1$.

*NdC: C est mis pour Carry (report).

Étape 2 — Mise en équation

En général, les tables de Karnaugh sont le moyen le plus commode pour trouver les simplifications. Dans les problèmes ultérieurs, nous sauterons éventuellement l'étape 1 pour considérer immédiatement les tables de Karnaugh. Il faut une table de Karnaugh pour chaque variable de sortie, soit:

B \ A	0	1
0	0	1
1	1	0

$$\sum = A\bar{B} + \bar{A}B$$

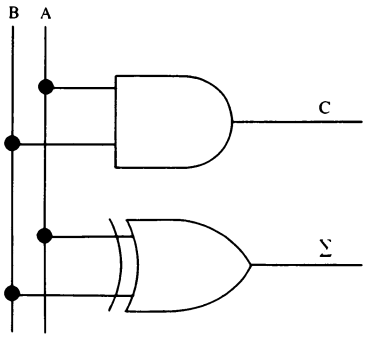
B \ A	0	1
0	0	0
1	0	1

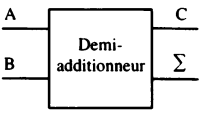
$$C = AB$$

Étape 3 — Implantation

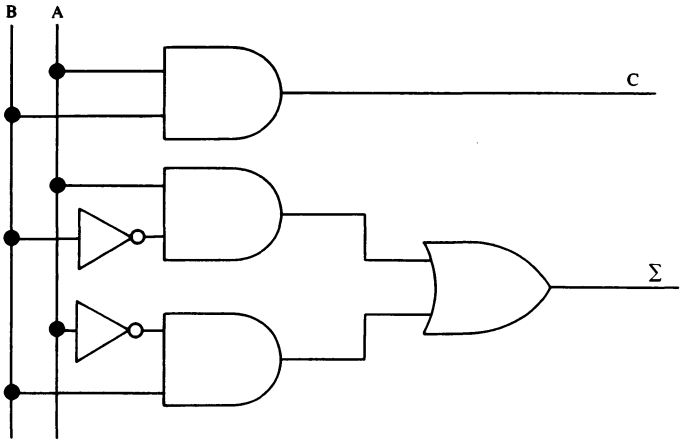
a) à l'aide de portes OU exclusif et ET.

$$\sum = A\bar{B} + \bar{A}B$$
$$= A \oplus B$$



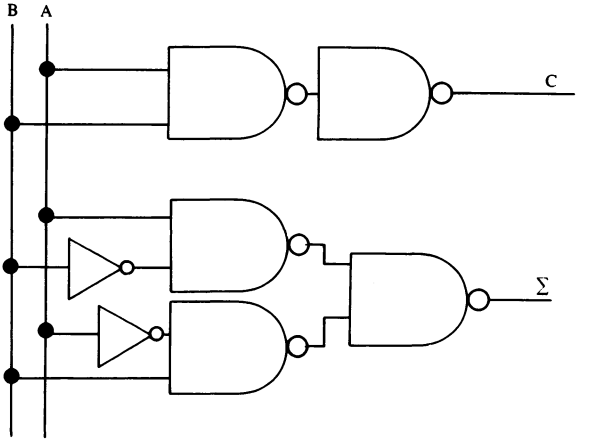


b) à l'aide d'inverseurs et de portes ET, OU



c) à l'aide d'inverseurs et de portes NON-ET.

On a: $\sum A\bar{B} + \bar{A}B = \overline{\overline{A\bar{B}} + \overline{\bar{A}B}} = \overline{\overline{A\bar{B}} \cdot \overline{\bar{A}B}}$ et $C = \overline{\overline{AB}}$



L'implantation dépend donc des circuits intégrés standard disponibles.

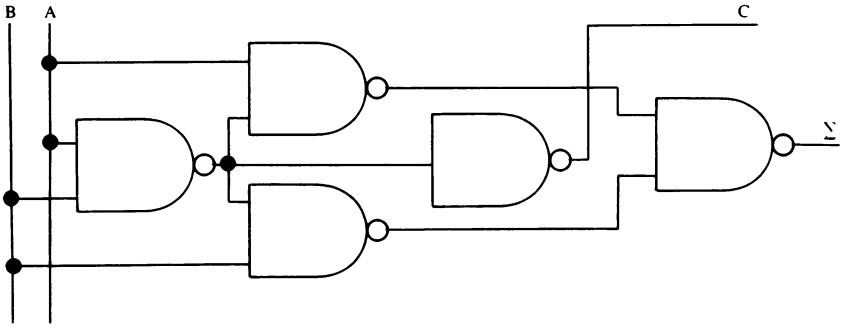
Une étude un peu plus poussée des fonctions à implanter permet parfois de les simplifier. Pour l'exemple traité, remarquons que:

$$A\bar{B} = A(\bar{A} + \bar{B}) = A\,\overline{AB}$$

$$\bar{A}B = B(\bar{A} + \bar{B}) = B\,\overline{AB}$$

alors $\sum = A\bar{B} + \bar{A}B = \overline{A\,\overline{AB}} \cdot \overline{B\,\overline{AB}}$ et $C = \overline{\overline{AB}}$

d'où l'implantation plus simple (pas d'inverseurs) suivante:



Si les variables $\bar{A}$ et $\bar{B}$ sont disponibles, ce qui est généralement le cas, alors les deux implantations ci-dessus sont de même complexité.

PROBLÈME 2 — DEMI-ADDITIONNEUR (AVEC RELAIS)

Construction d'un demi-additionneur à l'aide de boutons-poussoirs et de relais.

Soit le problème suivant:

Deux ouvriers travaillent sur une même pièce mécanique posée sur un tapis roulant. Ils ont chacun à leur disposition, de chaque côté du tapis, un bouton-poussoir qu'ils maintiennent enfoncé lorsque, leur travail sur cette pièce terminé, ils veulent amener la suivante devant eux. Lorsqu'un des ouvriers appuie son bouton-poussoir, un lampe s'allume. Si les deux appuient sur leur bouton-poussoir respectif, le tapis roulant se met en marche et la lampe s'éteint. Si l'un deux relâche son bouton, le tapis s'arrête pour des raisons de sécurité et la lampe s'allume. Si les deux relâchent leur bouton-poussoir respectif, la lampe s'éteint.

Mettre ce problème d'automatisme en équation et le résoudre.

Solution

Étape 1 — Mise en équation de la commande de la lampe L et du moteur du tapis M.

Désignons les boutons-poussoirs qui sont les variables d'entrée par A et B.

Table de Karnaugh de la lampe:

B \ A	0	1
0	0	1
1	1	0

$$L = A\bar{B} + \bar{A}B$$

La lampe s'allume si un seul bouton-poussoir est enfoncé.

Table de Karnaugh du moteur:

B \ A	0	1
0	0	0
1	0	1

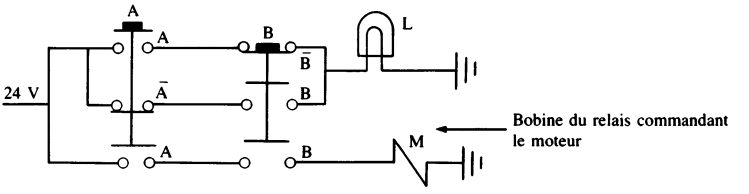
$$M = AB$$

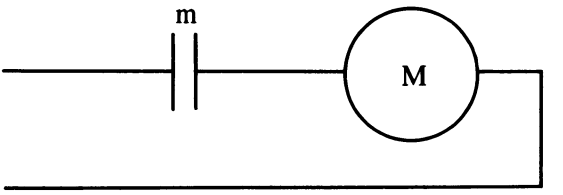
Le moteur ne se met en marche que si les deux boutons-poussoirs sont enfoncés.

On reconnaît les équations d'un demi-additionneur avec

$$L = \sum \text{ et } M = C.$$

Étape 2 — Implantation à l'aide d'un relais et de boutons-poussoirs

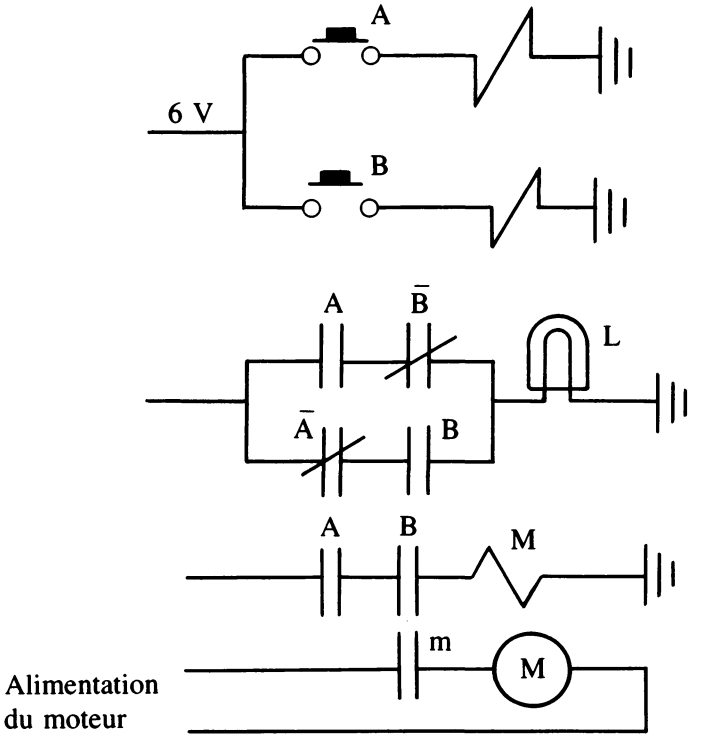




Pour des raisons de sécurité, la tension de commande est beaucoup plus petite que celle de l'alimentation du moteur,

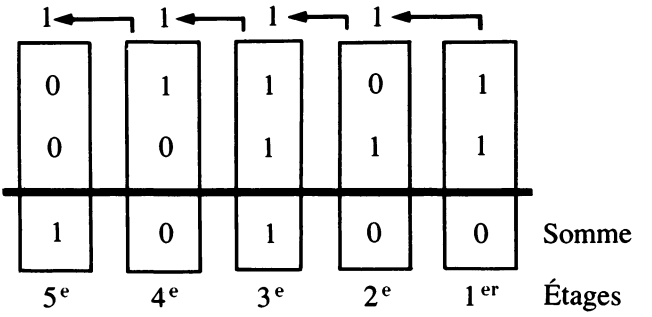
Autre implantation

Les boutons-poussoirs commandent les contacts de relais.

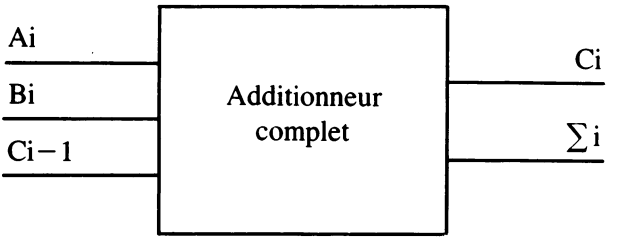


PROBLÈME 3 — ADDITIONNEUR COMPLET

Pour additionner deux nombres binaires, il faut tenir compte du report de l'étage précédent. Exemple de reports transférés aux étages suivants:



Il faut donc concevoir un circuit à trois entrées: les entrées Ai et Bi de l'étage i considéré et l'entrée Ci-1 du report de l'étage i considéré et l'entrée Ci − 1 du report de l'étage précédent i − 1 et deux sorties: la somme $\sum$i et le report Ci.



Solution

Étape 1 — Table de vérité

Entrées			Sorties	
Ci−1	Bi	Ai	$\sum$i	Ci
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Étape 2 — Mise en équation

Table de Karnaugh de $\sum$i

Ci−1 \ BiAi	00	01	11	10	
0	0	1	0	1	$\sum$i
1	1	0	1	0	

$$\sum i = Ai\overline{Bi}\,\overline{Ci-1} + \overline{Ai}Bi\,\overline{Ci-1} + \overline{Ai}\,\overline{Bi}\,Ci-1 + AiBi\,Ci-1$$
$$= (Ai\overline{Bi} + \overline{Ai}Bi)\,\overline{Ci-1} + (\overline{Ai}\,\overline{Bi} + AiBi)\,Ci-1$$
$$= (Ai \oplus Bi)\,\overline{Ci-1} + \overline{Ai \oplus Bi}\;\; Ci-1$$
$$= (Ai \oplus Bi) \oplus Ci-1$$

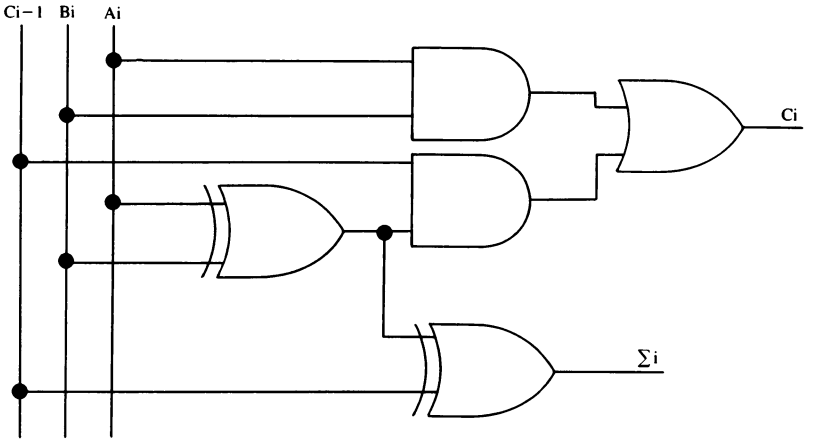
Table de Karnaugh de Ci

Ci−1 \ BiAi	00	01	11	10
0	0	0	1	0
1	0	1	1	1

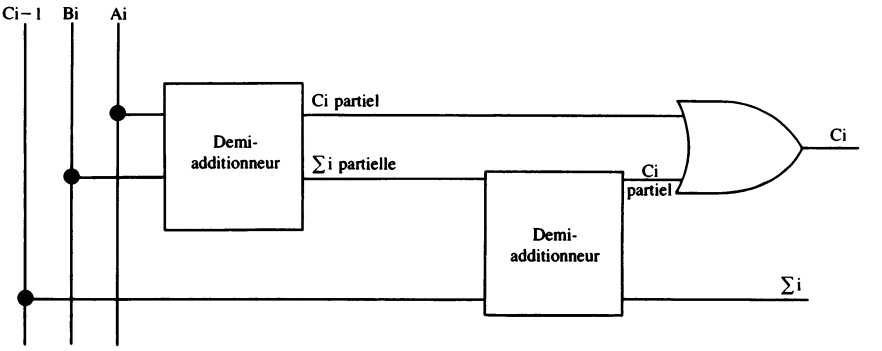
Ci

$$Ci = AiBi + Ai\overline{Bi}\,Ci-1 + \overline{Ai}Bi\,Ci-1$$
$$= AiBi + (Ai \oplus Bi)\,Ci-1$$

Étape 3 — Implantation



Un tel montage constitué de deux demi-additionneurs montés selon la configuration suivante est appelé un additionneur complet.



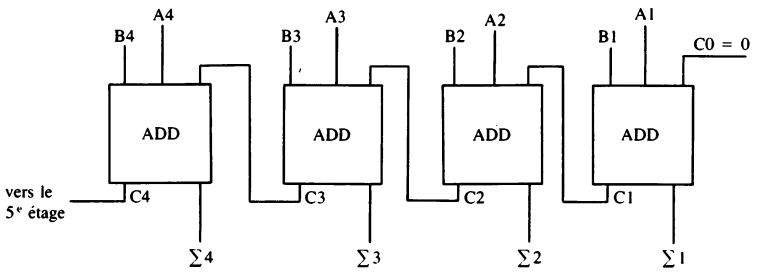
Structure d'un additionneur complet

Cette configuration permet d'analyser plus facilement l'additionneur complet. La demi-addition des deux bits d'un étage quelconque donne une somme et un report partiels. La demi-addition du report de l'étage précédent et de cette somme partielle donne, d'une part, la somme de l'étage sur lequel on travaille et, d'autre part, une second report partiel qui, combiné avec le premier à l'aide d'une porte OU, donne le report Ci de l'étage envisagé.

Considérons l'addition de deux nombres binaires de quatre bits:

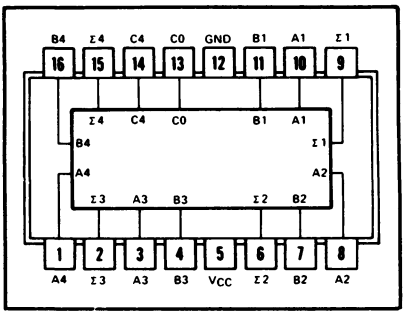
$$\begin{array}{rcccc} & A4 & A3 & A2 & A1 \\ + & B4 & B3 & B2 & B1 \end{array}$$

On aura le montage:

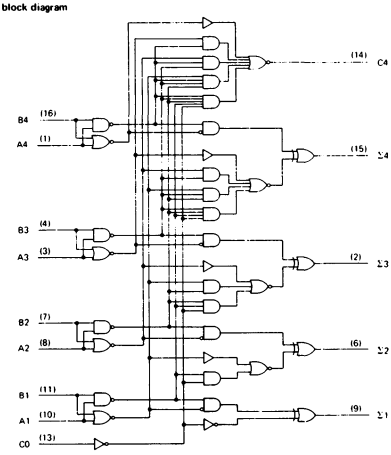


Le circuit intégré 7483 implante une telle fonction, son schéma de brochage apparaît ci-dessous. On peut donc par combinaison de CI 7483 additionner deux, trois, . . . nombres binaires quelconques.

Les CI 7480 et 7482 sont des additionneurs respectivement de 1 bit et 2 bits.



functional block diagram



TABLE

INPUT				OUTPUT: WHEN C0 = L			OUTPUT: WHEN C0 = H		
						WHEN C2 = L			WHEN C2 = H
A1 / A3	B1 / B3	A2 / A4	B2 / B4	Σ1 / Σ3	Σ2 / Σ4	C2 / C4	Σ1 / Σ3	Σ2 / Σ4	C2 / C4
L	L	L	L	L	L	L	H	L	L
H	L	L	L	H	L	L	L	H	L
L	H	L	L	H	L	L	L	H	L
H	H	L	L	L	H	L	H	H	L
L	L	H	L	L	H	L	H	H	L
H	L	H	L	H	H	L	L	L	H
L	H	H	L	H	H	L	L	L	H
H	H	H	L	L	L	H	H	L	H
L	L	L	H	L	H	L	H	H	L
H	L	L	H	H	H	L	L	L	H
L	H	L	H	H	H	L	L	L	H
H	H	L	H	L	L	H	H	L	H
L	L	H	H	L	L	H	H	L	H
H	L	H	H	H	L	H	L	H	H
L	H	H	H	H	L	H	L	H	H
H	H	H	H	L	H	H	H	H	H

Additionneur complet de quatre bits, selon le CI 7483.

PROBLÈME 4 — DEMI-SOUSTRACTEUR

Concevoir un circuit capable de soustraire un bit d'un autre bit, capable donc de générer leur différence D et leur retenue C*.

Solution

Soit A et B ces deux bits. Nous désirons obtenir la différence D = A moins B et la retenue C.

*NdC. Nous conserverons la même lettre que pour le report de l'addition.

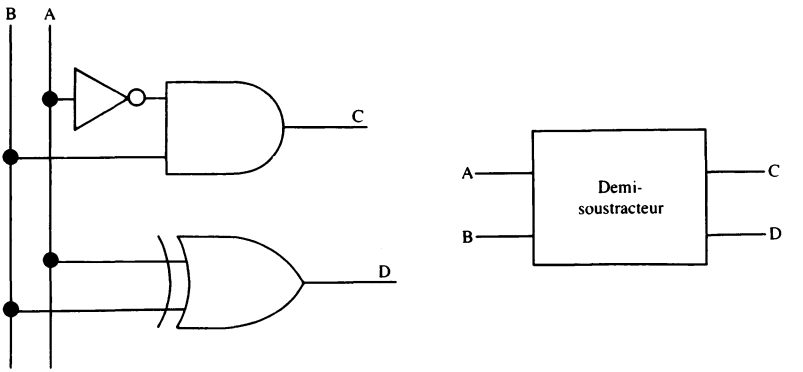
Entrées		Sorties	
B	A	D	C
0	0	0	0
0	1	1	0
1	0	1	1
1	1	0	0

Étape 2— Mise en équation

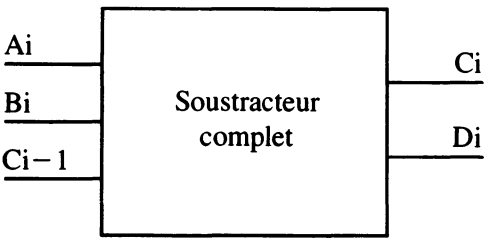
Nous pouvons écrire immédiatement les équations du circuit à partir de la table de vérité. On obtient alors:

$$D = A\bar{B} + \bar{A}B$$
$$= A \oplus B$$
$$C = \bar{A}B$$

Étape 3 — Implantation



PROBLÈME 5 — SOUSTRACTEUR COMPLET



On veut avoir la différence Di de deux bits, Ai moins Bi, en tenant compte de la retenue Ci−1 de l'étage précédent i−1 et générer la retenue Ci pour l'étage suivant.

Solution

Étape 1 — Table de vérité

Entrées			Sorties	
Ci−1	Bi	Ai	Di	Ci
0	0	0	0	0
0	0	1	1	0
0	1	0	1	1
0	1	1	0	0
1	0	0	1	1
1	0	1	0	0
1	1	0	0	1
1	1	1	1	1

Étape 2 — Mise en équation

Table de Karnaugh de Di

Ci−1 \ BiAi	00	01	11	10
0	0	1	0	1
1	1	0	1	0

Di

$$Di = (Ai\overline{Bi} + \overline{Ai}Bi)\,\overline{Ci-1} + (\overline{Ai}\,\overline{Bi} + AiBi)\,Ci-1$$

$$= (Ai \oplus Bi)\,\overline{Ci\text{-}1} + \overline{Ai \oplus Bi}\;\; Ci-1$$

$$= Ai \oplus Bi \oplus Ci-1$$

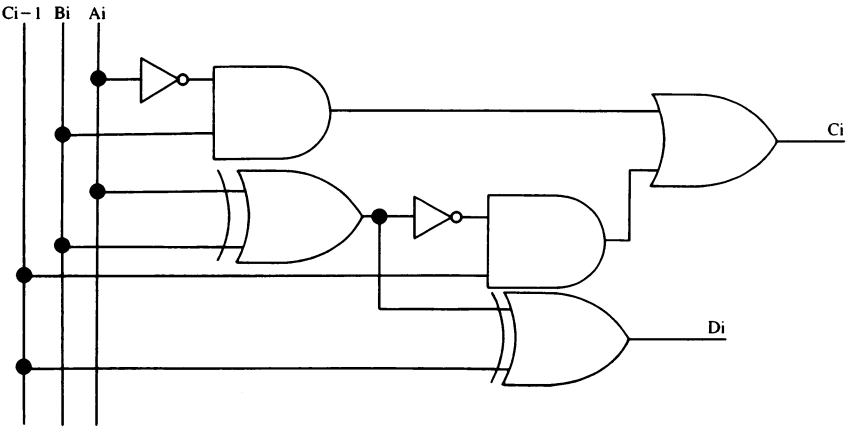
Table de Karnaugh de Ci

Ci−1 \ BiAi	00	01	11	10
0	0	0	0	1
1	1	0	1	1

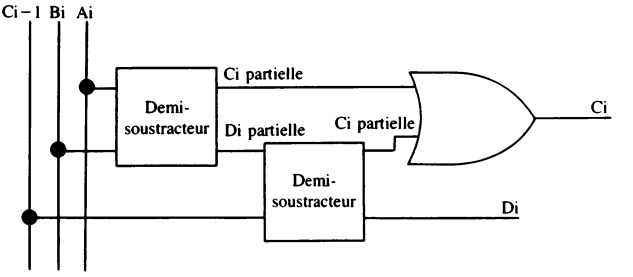
Ci

$$Ci = \overline{Ai}Bi + (\overline{Ai}\overline{Bi} + AiBi)\, Ci-1$$
$$= \overline{Ai}Bi + \overline{Ai \oplus Bi}\ \ Ci-1$$

Étape 3 — Implantation



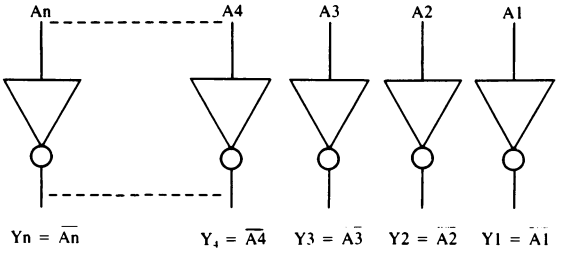
De la même façon que l'additionneur complet est constitué de deux demi-additionneurs, le soustracteur complet comprend deux demi-soustracteurs.



Structure d'un soustracteur complet

PROBLÈME 6 — COMPLÉMENTEUR À 1 ET APPLICATION À LA SOUSTRACTION

Complémenter un nombre binaire à 1 revient à inverser chacun de ses bits. Il suffit donc de coupler un inverseur à chaque bit de ce nombre binaire.



Nombre donne: An - - -A4 A3 A2 A1

Nombre complémenté à 1:

$$Yn\text{- - - }Y4\ Y3\ Y2\ Y1 = \overline{An}\text{ - - - }\overline{A4}\ \overline{A3}\ \overline{A2}\ \overline{A1}$$

On sait aussi qu'on peut ramener une soustraction à une addition en prenant le complément à 1 du diminuteur, en l'additionnant au diminuende et en ajoutant ensuite 1 au résultat. On ne fera la complémentation que lorsqu'on a une soustraction. Il nous faut donc un circuit qui ne fasse le complément que lorsqu'un signal de commande C est égal à 1 pour la première implantation ci-dessous, à 0 pour la seconde (avec B = 0) et laisse le nombre tel quel si C est égal à 0, pour la première implantation, à 1 pour la seconde (avec B = 0).

Solution

Étape 1 — Table de vérité

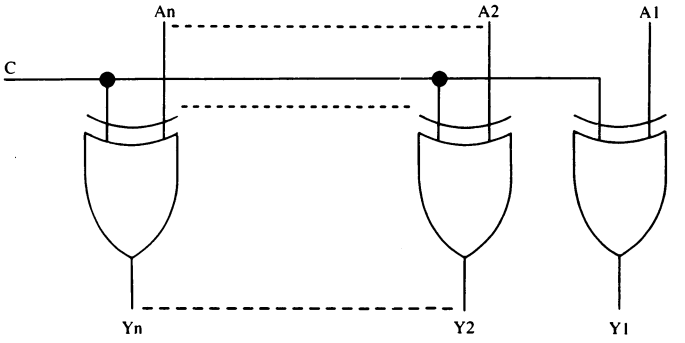
Entrées		Sortie
C	An	Yn
0	0	0
0	1	1
1	0	1
1	1	0

Étape 2 — mise en équation

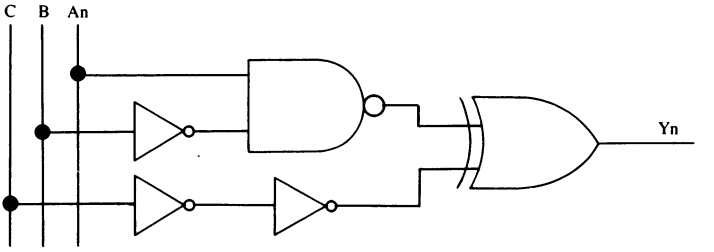
$$Yn = An\overline{C} + \overline{An}C$$
$$= An \oplus C$$

Étape 3 — Implantation

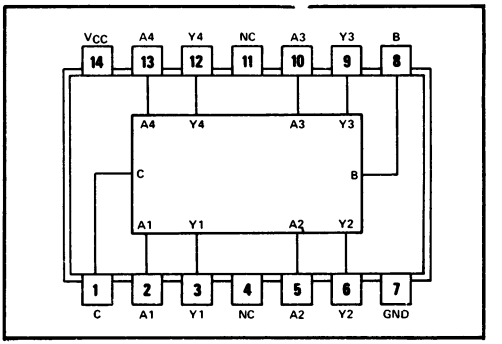
a) à l'aide de CI 7486 comprenant quatre portes OU exclusif à deux entrées.



b) à l'aide de Ci 7487 à quatre bits.



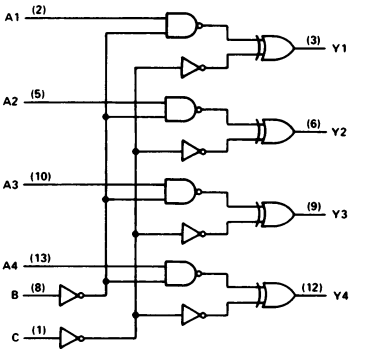
Soit B = 0, si C = 0 alors la sortie est le complément à 1 du nombre de quatre bits à l'entrée, si C = 1 alors l'entrée passe telle quelle à la sortie. Nous verrons une utilisation de ce circuit plus loin. Soit B = 1, si C = 1 on ne complémente pas puisque toutes les sorties de ce circuit valent 0, si C = 0, on n'a que des 1 à la sortie. Le complément à 2 de 1 étant uniquement constitué de 1 si on effectue la soustraction, dans ce cas, le nombre à la sortie diminuera de 1. Cela peut être utile dans certains cas.



FUNCTION TABLE

CONTROL INPUTS		OUTPUTS			
B	C	Y1	Y2	Y3	Y4
L	L	$\overline{A1}$	$\overline{A2}$	$\overline{A3}$	$\overline{A4}$
L	H	A1	A2	A3	A4
H	L	H	H	H	H
H	H	L	L	L	L

FUNCTIONAL BLOCK DIAGRAM



CI 7487 à quatre bits à sortie complémentée vraie, 1 ou 0

PROBLÈME 7 — COMPLÉMENTEUR À 2

Implanter le complémenteur à 2 d'un nombre de quatre bits.

Solution

Étape 1 — Table de vérité

A4	A3	A2	A1	Y4	Y3	Y2	Y1
0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1
0	0	1	0	1	1	1	0
0	0	1	1	1	1	0	1
0	1	0	0	1	1	0	0
0	1	0	1	1	0	1	1
0	1	1	0	1	0	1	0
0	1	1	1	1	0	0	1
1	0	0	0	1	0	0	0
1	0	0	1	0	1	1	1
1	0	1	0	0	1	1	0
1	0	1	1	0	1	0	1
1	1	0	0	0	1	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	0	1	0
1	1	1	1	0	0	0	1

Étape 2 — Mise en équation

a) Y1

A4A3 \ A2A1	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	0	1	1	0
10	0	1	1	0

Y1

$$Y1 = A1$$

b) Y2

A4A3 \ A2A1	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

Y2

$$Y2 = \overline{A2}A1 + A2\overline{A1}$$
$$= A2 \oplus A1$$

c) Y3

A4A3 \ A2A1	00	01	11	10
00	0	1	1	1
01	1	0	0	0
11	1	0	0	0
10	0	1	1	1

Y3

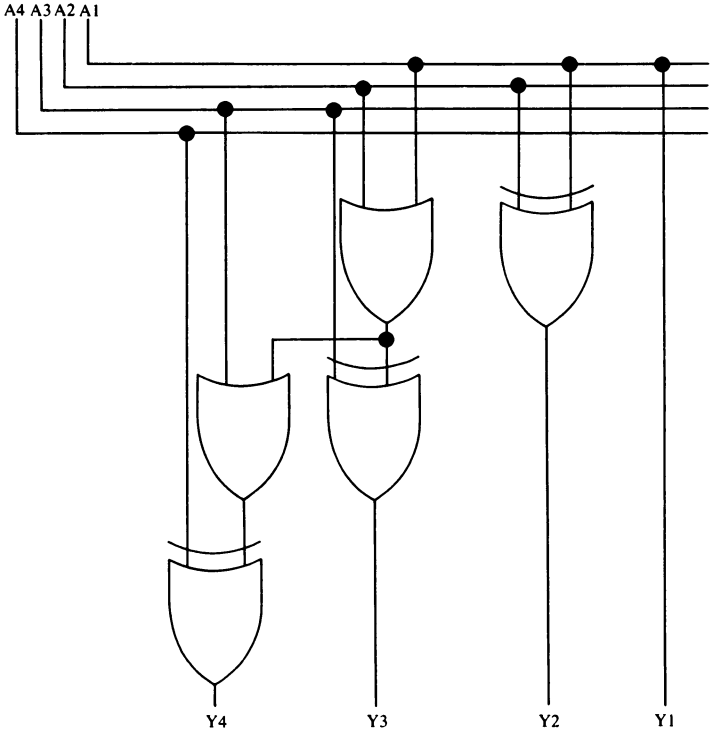
$$Y3 = \overline{A3}\,(A2 + A1) + A3\,\overline{A2}\,\overline{A1}$$
$$= \overline{A3}\,(A2 + A1) + A3\,\overline{A2 + A1}$$
$$= A3 \oplus (A2 + A1)$$

d) Y4

A4A3 \ A2A1	00	01	11	10
00	0	1	1	1
01	1	1	1	1
11	0	0	0	0
10	1	0	0	0

Y4

$$Y4 = \overline{A4}\,(A2 + A1) + \overline{A4}\,A3 + A4\,\overline{A3}\,\overline{A2}\,\overline{A1}$$
$$= \overline{A4}\,(A3 + A2 + A1) + A4\,\overline{A3 + A2 + A1}$$
$$= A4 \oplus (A3 + A2 + A1)$$



Ce type de circuit se généralise. On peut donc construire le circuit complémenteur à 2 de n'importe quel nombre binaire.

PROBLÈME 8 — ADDITIONNEUR — SOUSTRACTEUR

Utilisation de l'additionneur intégré à quatre bits comme additionneur-soustracteur et du circuit *vrai complément*.

Nous avons vu au chapitre que les nombres négatifs peuvent être représentés par leur complément à 1 ou leur complément à 2.

Le complément à 1 d'un nombre s'obtient en inversant chaque bit de ce nombre. Le complément à 2 s'obtient en ajoutant 1 au complément à 1 du nombre considéré.

Considérons la représentation des nombres avec un bit signe (celui de gauche). S'il est égal à 0, le nombre est positif; s'il est égal à 1, le nombre est négatif. Rappelons l'algorithme de l'addition et celui de la soustraction.

1er cas: complément à 1

Prenons l'exemple des nombres 16 et 13 avec huit bits.

	signe	valeur binaire
+16	0	0010000
+13	0	0001101

Considérons les différentes combinaisons de signe, on aura:

+16	00010000	+16	00010000
+13	00001101	−13	11110010
29	00011101		1 00000010 (1 → 1)
		+ 3	00000011

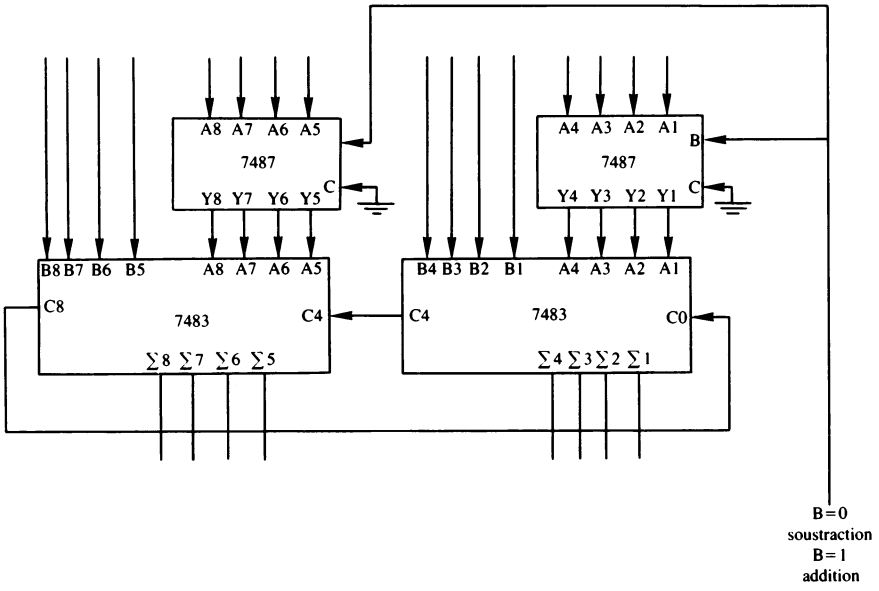
−16	11101111	
+13	00001101	
− 3	11111100	dont le complément à 1 est 00000011, soit 3

−16	11101111	
−13	11110010	
	1 11100001 (1 → 1)	
−29	11100010	dont le complément à 1 est 00011101, soit 29.

De ces quatre cas on peut tirer les règles générales suivantes:

a) effectuer l'addition de deux nombres positifs en incluant leur bit signe.

b) pour la soustraction, complémenter à 1 le nombre à soustraire et additionner en incluant les bits signes. S'il y a un débordement, l'ajouter au résultat. Ces deux opérations se ramènent donc à une addition. Le schéma de l'additionneur-soustracteur donné ci-dessous convient à des nombres de huit bits, incluant le bit signe. Ce schéma peut se généraliser à n'importe quel nombre de bits. Le dernier report (débordement) est ramené à l'entrée report du premier étage pour son addition si nécessaire.



2[e] cas: complément à 2

Reprenons les exemples du premier cas.

+16	00010000	+16	00010000	−16	11110000
+13	00001101	−13	11110011	+13	00001101
+29	00011101	+ 3	1 00000011	− 3	11111101

débordement à éliminer

00000011 en complément à 2, soit 3

-16 11110000

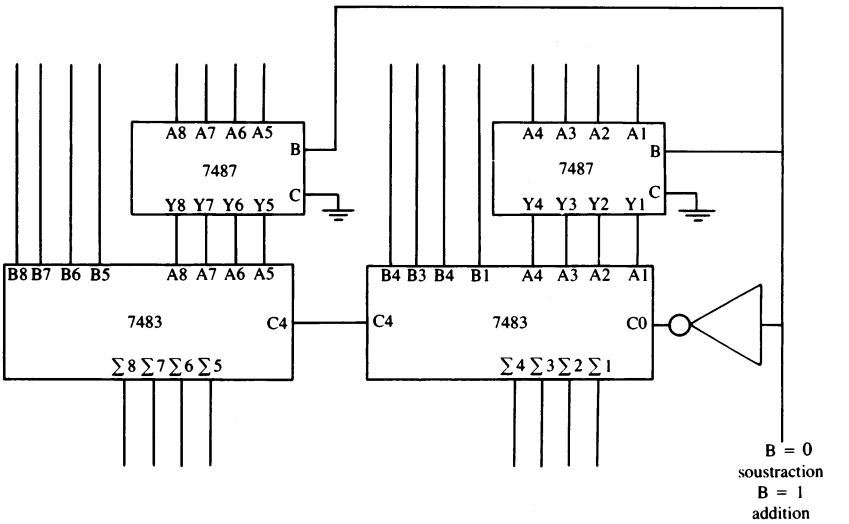
-13 11110011

-29 [1] 11100011 = 00011101 en complément à 2, soit 29

débordement
à éliminer

Pour la soustraction, on additionne le complément à 2 du nombre à soustraire à l'autre nombre en incluant les bits signes. On ignore toujours le dernier report, s'il existe.

Le schéma de cet additionneur-soustracteur est donné ci-dessous. Le complément à 2 se fait comme dans le cas précédent avec le CI 74487 de complément à 1 et on ajoute 1 par l'entrée de report C0 du premier étage. Les nombre sont de huit bits, incluant le bit signe.



PROBLÈME 9 — CONVERTISSEUR GRAY — BINAIRE NATUREL

Convertir le code Gray de quatre bits en code binaire.

Solution

Étape 1 — Table de vérité

Code décimal	Entrées code Gray				Sorties code binaire naturel			
	G4	G3	G2	G1	B4	B3	B2	B1
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	1	0	0	1	0
3	0	0	1	0	0	0	1	1
4	0	1	1	0	0	1	0	0
5	0	1	1	1	0	1	0	1
6	0	1	0	1	0	1	1	0
7	0	1	0	0	0	1	1	1
8	1	1	0	0	1	0	0	0
9	1	1	0	1	1	0	0	1
10	1	1	1	1	1	0	1	0
11	1	1	1	0	1	0	1	1
12	1	0	1	0	1	1	0	0
13	1	0	1	1	1	1	0	1
14	1	0	0	1	1	1	1	0
15	1	0	0	0	1	1	1	1

Étape 2 — Mise en équation

a) B4

G4G3 \ G2G1	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

B4

B4 = G4

b) B3

G4G3 \ G2G1	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

B3

$$B3 = \overline{G4}\ G3 + G4\ \overline{G3}$$
$$= G4 \oplus G3$$

c) B2

G4G3 \ G2G1	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	0	0	1	1
10	1	1	0	0

B2

$$B2 = \overline{G4}\ \overline{G3}\ G2 + \overline{G4}\ G3\ \overline{G2} + G4\ G3\ G2 + G4\ \overline{G3}\ \overline{G2}$$
$$= (\overline{G4}\ \overline{G3} + G4\ G3)\ G2 + (\overline{G4}\ G3 + G4\ \overline{G3})\ \overline{G2}$$
$$= \overline{G4 \oplus G3}\ G2 + (G4 \oplus G3)\ G2$$
$$= G4 \oplus G3 \oplus G2$$

d) B1

G4G3 \ G2G1	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0

B1

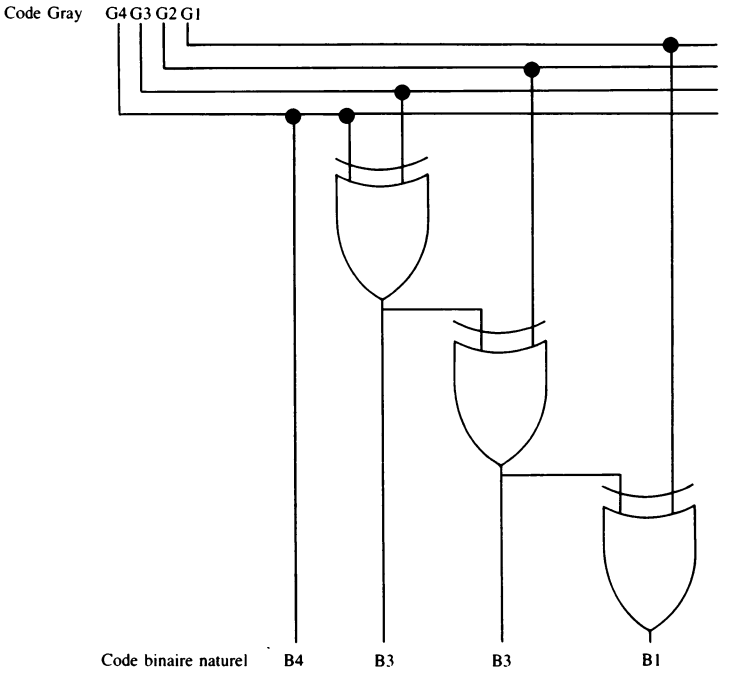
$$B1 = \overline{G4}\,\overline{G3}\,(\overline{G2}\,G1 + G2\,\overline{G1}) + \overline{G4}\,G3\,(\overline{G2}\,\overline{G1} + G2\,G1) + G4\,G3\,(\overline{G2}\,G1 + G2\,\overline{G1}) + G4\,\overline{G3}\,(\overline{G2}\,\overline{G1} + G2\,G1)$$

$$= (\overline{G4}\,\overline{G3} + G4\,G3)(\overline{G2}\,G1 + G2\,\overline{G1}) + (\overline{G4}\,G3 + G4\,\overline{G3})(\overline{G2}\,\overline{G1} + G2\,G1)$$

$$= \overline{G4 \oplus G3}\,(G2 \oplus G1) + (G4 \oplus G3)\,\overline{G2 \oplus G1}$$

$$= G4 \oplus G3 \oplus G2 \oplus G1$$

Étape 3 — Implantation



Ce circuit peut se généraliser pour convertir n bits de code Gray en n bits de code binaire.

REMARQUE La manipulation des circuits OU exclusifs permet d'obtenir quelquefois des schémas d'implantation fort simples.

Problème Convertir un code binaire en code Gray.

On va se servir du problème précédent et de sa solution: on voit immédiatement que G4 = B4.

Par ailleurs:

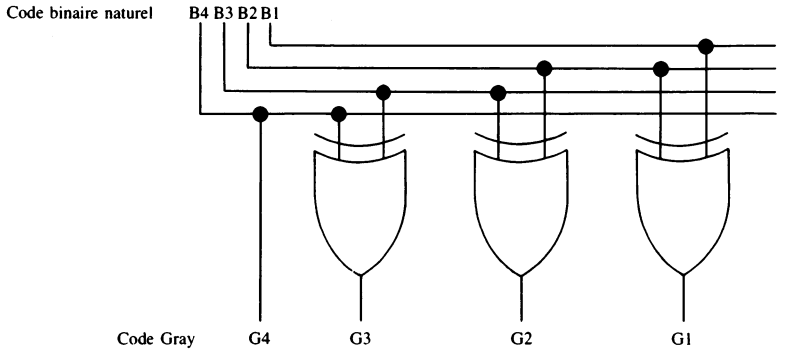
$$B4 \oplus B3 = G4 \oplus G4 \oplus G3$$
$$= G3 \quad \text{car } G4 \oplus G4 = \overline{G4}\,G4 + G4\,\overline{G4}$$
$$= 0$$

soit donc: $G3 = B4 \oplus B3$

On démontre de la même façon que

$$G2 = B3 \oplus B2$$
$$G1 = B2 \oplus B1$$

d'où l'implantation.



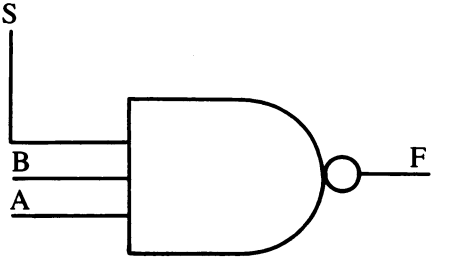
PROBLÈME 10 — UNITÉ ARITHMÉTIQUE ET LOGIQUE

Nous avons vu ci-dessus comment concevoir des circuits simples de logique combinatoire. Étudions maintenant un circuit combinatoire qui nous permettra d'effectuer un plus grand nombre d'opérations ou d'implanter des fonctions choisies par un code binaire appelé code d'opération par analogie avec les codes des ordinateurs. Ce type de circuit constitue le cerveau des ordinateurs.

Examinons tout d'abord deux cas simples pour illustrer le problème.

1 er cas:

Soit une porte NON-ET à trois entrées: une entrée S de commande et deux entrées A et B de données.



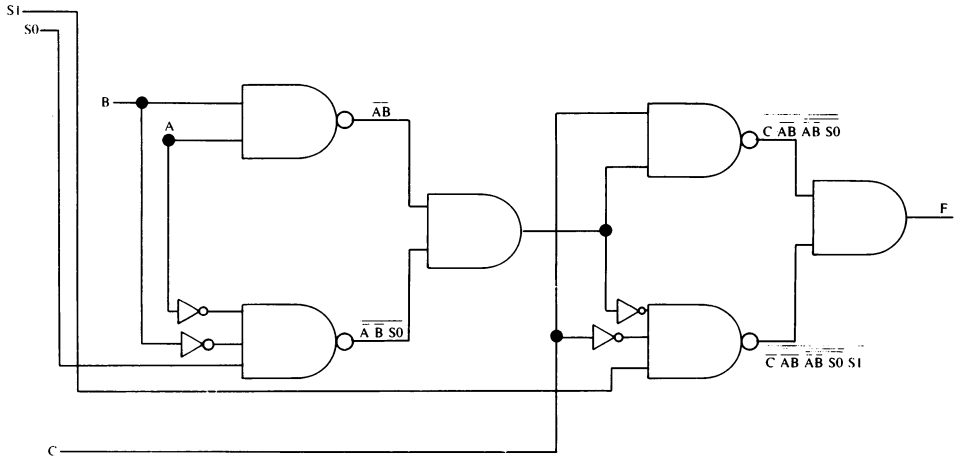
La table 1 ci-dessous énumère toutes les combinaisons possibles.

Table 1

S = 0	S = 1		
F = 1	B	A	F
	0	0	1
	0	1	1
	1	0	1
	1	1	0

2e cas:

Soit le circuit suivant. Analyser la fonction F de sortie suivant les valeurs de C, S0 et S1 et selon les valeurs de A, S0 et S1.



Solution Avec les fonctions partielles inscrites sur le schéma, on obtient:

$$F = \overline{C\,AB\,\overline{A}\,\overline{B}\,S0} \cdot \overline{C\,AB\,\overline{AB}S0\,S1}$$

1er cas: C = 0, alors

$$F = \overline{\overline{\overline{AB}\ \overline{\overline{AB}}S0}\ S1}$$

$$= \overline{(AB + \overline{\overline{AB}}S0)\ S1}$$

a) Si S1 = 0, alors $F = \overline{1}$

b) Si S1 = 1, alors $F = \overline{AB + \overline{\overline{AB}}\ S0}$

si, de plus, S0 = 0, alors $F = \overline{AB}$

ou S0 = 1, alors $F = \overline{AB + \overline{\overline{AB}}}$

$$= A \oplus B$$

2e cas: C = 1, alors

$$F = \overline{\overline{AB}\ \overline{\overline{AB}}\ S0}$$

a) si S0 = 0, alors F = AB

b) si S0 = 1, alors $F = AB + \overline{\overline{AB}}$

$$= \overline{A \oplus B}$$

d'où les tables ci-dessous.

C = 0

S1	S0	F
0	0	1
0	1	1
1	0	$\overline{AB}$
1	1	$A \oplus B$

C = 1

S1	S0	F
0	0	$\overline{AB}$
0	1	$\overline{A \oplus B}$
1	0	$\overline{AB}$
1	1	$\overline{A \oplus B}$

Analysons maintenant F en fonction de A, S0 et S1.

$$F = \overline{C\ \overline{\overline{AB}}\ \overline{\overline{AB}}\ S0} \cdot \overline{\overline{C}\ \overline{\overline{AB}}\ \overline{\overline{\overline{AB}}S0}\ S1}$$

1er cas: A = 0, alors

$$F = \overline{C\,\overline{B}\,S0} \cdot \overline{\overline{C}\,\overline{B}\,S0\,S1}$$

a) Si S0 = 0, alors $F = \overline{C}$

b) Si S0 = 1, alors $F = \overline{BC} \cdot \overline{\overline{C}\overline{B}\,S1}$

si, de plus, S1 = 0, alors $F = \overline{BC}$

ou S1 = 1, alors $F = \overline{BC} \cdot \overline{\overline{B}\overline{C}}$

$$= \overline{BC + \overline{B}\overline{C}}$$

$$= B \oplus C$$

2e cas: A = 1, alors

$$F = \overline{C\overline{B}} \cdot \overline{\overline{C}B\,S1}$$

a) Si S1 = 0, alors $F = \overline{C\overline{B}}$

$$= B + \overline{C}$$

b) Si S1 = 1, alors $F = \overline{C\overline{B} + \overline{C}B}$

$$= \overline{B \oplus C}$$

d'où les tables ci-dessous.

A = 0

S1	S0	F
0	0	$\overline{C}$
0	1	$\overline{BC}$
1	0	$\overline{C}$
1	1	$B \oplus C$

A = 1

S1	S	F
0	0	$B + \overline{C}$
0	1	$\overline{B + \overline{C}}$
1	0	$\overline{B \oplus C}$
1	1	$B \oplus C$

Cette idée d'un composant programmable a été développée et a débouché sur le circuit intégré ALU 74181* qui est l'optimum pour deux nombres de quatre bits. Les figures suivantes en donnent le schéma de brochage, le *diagramme logique* et la *table de fonctions* pour une logique positive (niveau haut égal à 1).

* ALU, mis pour *Arithmetic and Logical Unit:* unité arithmétique et logique.

Nous allons étudier plus en détail le *diagramme électrique* des fonctions logiques du circuit intégré ALU 74181 pour M = 1. Voir sa représentation plus bas et nous vérifierons une colonne de la table 2 des fonctions générées, celle des fonctions logiques. Voir ci-dessous.

Nous voyons que la fonction logique à la sortie est:

$$Fi = Xi\ Yi + \overline{Xi}\ \overline{Yi}$$

Avec

$$Xi = \overline{Ai\ \overline{Bi}\ S2 + Ai\ Bi\ S3}$$

$$= \overline{Ai\ (\overline{Bi}\ S2 + Bi\ S3)}$$

$$= \overline{Ai} + \overline{\overline{Bi}\ S2 + Bi\ S3}$$

$$= \overline{Ai} + \overline{\overline{Bi}\ S2}\quad \overline{Bi\ S3}$$

$$= \overline{Ai} + (Bi + \overline{S2})\ (\overline{Bi} + \overline{S3})$$

$$= \overline{Ai} + Bi\ \overline{S3} + \overline{Bi}\ \overline{S2} + \overline{S3}\ \overline{S2}$$

et

$$Yi = \overline{Ai + Bi\ S0 + \overline{Bi}\ S1}$$

$$= \overline{Ai}\quad \overline{Bi\ S0}\quad \overline{\overline{Bi}\ S1}$$

$$= Ai\ (\overline{Bi} + \overline{S0})\ (Bi + \overline{S1})$$

$$= Ai\ (\overline{Bi}\ \overline{S1} + Bi\ \overline{S0} + \overline{S1}\ \overline{S0})$$

Nous voyons que Xi ne dépend que de Ai, Bi, S3 et S2, nous pouvons donc dresser la table suivante:

S3	S2	Xi
0	0	1
0	1	$\overline{Ai} + Bi$
1	0	$Ai + \overline{Bi}$
1	1	$\overline{Ai}$

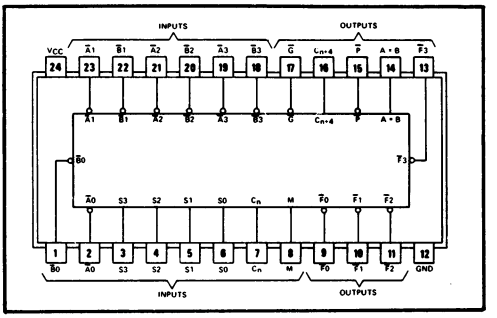
De même Yi ne dépend que de Ai, Bi, S1 et S0, nous avons donc la table de fonctionnement suivante:

S1	S0	Yi
0	0	$\overline{Ai}$
0	1	$\overline{Ai}\,\overline{Bi}$
1	0	$\overline{Ai}\,Bi$
1	1	0

Nous pouvons alors établir la table suivante dans laquelle les indices i sont sous-entendus d'office pour ne pas surcharger les écritures.

S3	S2	S1	S0	X	Y	XY	$\overline{X}\,\overline{Y}$	$F = XY + \overline{X}\,\overline{Y}$
0	0	0	0	1	$\overline{A}$	$\overline{A}$	0	$\overline{A}$
0	0	0	1	1	$\overline{A}\,\overline{B}$	$\overline{A}\,\overline{B}$	0	$\overline{A + B}$
0	0	1	0	1	$\overline{A}B$	$\overline{A}B$	0	$\overline{A}B$
0	0	1	1	1	0	0	0	0
0	1	0	0	$\overline{A} + B$	$\overline{A}$	$\overline{A}$	$A\overline{B}$	$\overline{AB}$
0	1	0	1	$\overline{A} + B$	$\overline{A}\,\overline{B}$	$\overline{A}\,\overline{B}$	$A\overline{B}$	$\overline{B}$
0	1	1	0	$\overline{A} + B$	$\overline{A}B$	$\overline{A}B$	$A\overline{B}$	$A \oplus B$
0	1	1	1	$\overline{A} + B$	0	0	$A\overline{B}$	$A\overline{B}$
1	0	0	0	$\overline{A} + \overline{B}$	$\overline{A}$	$\overline{A}$	AB	$\overline{A} + B$
1	0	0	1	$\overline{A} + \overline{B}$	$\overline{A}\,\overline{B}$	$\overline{A}\,\overline{B}$	AB	$\overline{A \oplus B}$
1	0	1	0	$\overline{A} + \overline{B}$	$\overline{A}B$	$\overline{A}B$	AB	B
1	0	1	1	$\overline{A} + \overline{B}$	0	0	AB	AB
1	1	0	0	$\overline{A}$	$\overline{A}$	$\overline{A}$	A	1
1	1	0	1	$\overline{A}$	$\overline{A}\,\overline{B}$	$\overline{A}\,\overline{B}$	A	$A + \overline{B}$
1	1	1	0	$\overline{A}$	$\overline{A}B$	$\overline{A}B$	A	A + B
1	1	1	1	$\overline{A}$	0	0	A	A

Cette dernière colonne est bien celle des fonctions logiques de la table 2 des fonctions générées ci-après.



PIN DESIGNATIONS

DESIGNATION	PIN NOS.	FUNCTION
A3, A2, A1, A0	19, 21, 23, 2	WORD A INPUTS
B3, B2, B1, B0	18, 20, 22, 1	WORD B INPUTS
S3, S2, S1, S0	3, 4, 5, 6	FUNCTION-SELECT INPUTS
C_n	7	INV. CARRY INPUT
M	8	MODE CONTROL INPUT
F3, F2, F1, F0	13, 11, 10, 9	FUNCTION OUTPUTS
A = B	14	COMPARATOR OUTPUT
P	15	CARRY PROPAGATE OUTPUT
C_{n+4}	16	INV. CARRY OUTPUT
G	17	CARRY GENERATE OUTPUT
V_{CC}	24	SUPPLY VOLTAGE
GND	12	GROUND

TABLE 1

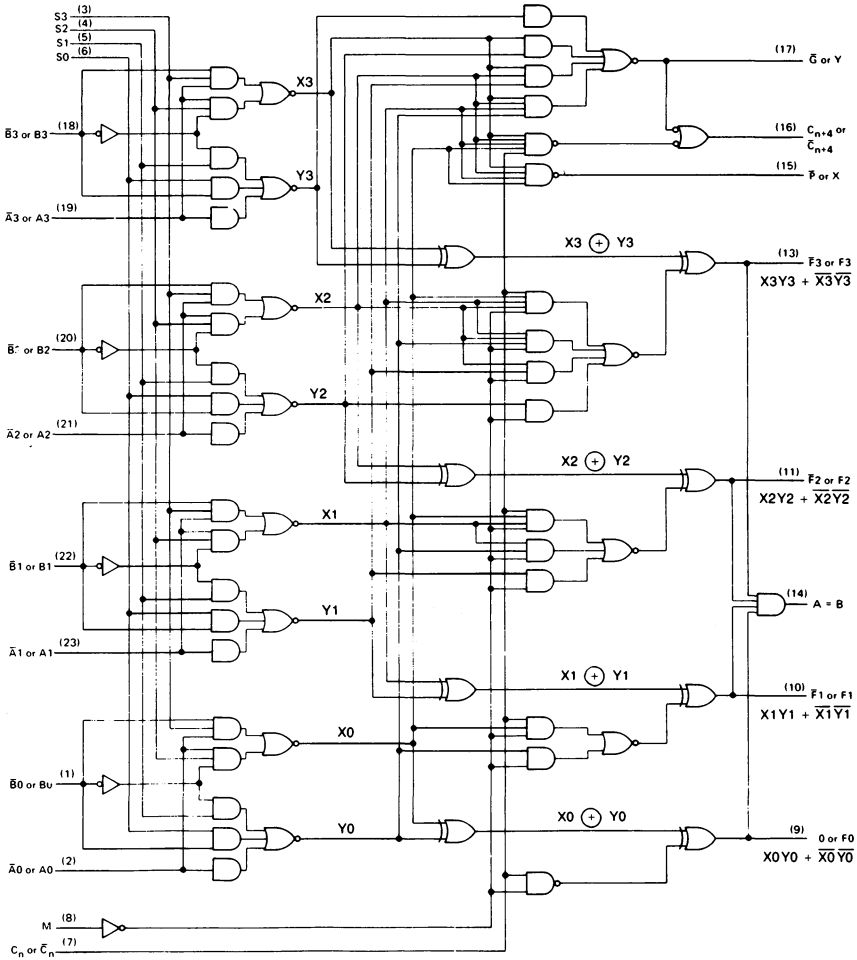
SELECTION				ACTIVE-LOW DATA		
				M = H	M = L; ARITHMETIC OPERATIONS	
S3	S2	S1	S0	LOGIC FUNCTIONS	C_n = L (no carry)	C_n = H (with carry)
L	L	L	L	F = $\overline{A}$	F = A MINUS 1	F = A
L	L	L	H	F = $\overline{A+B}$	F = AB MINUS 1	F = AB
L	L	H	L	F = $\overline{A}$ + B	F = A$\overline{B}$ MINUS 1	F = A$\overline{B}$
L	L	H	H	F = 1	F = MINUS 1 (2's COMP)	F = ZERO
L	H	L	L	F = $\overline{A+\overline{B}}$	F = A PLUS (A + $\overline{B}$)	F = A PLUS (A + $\overline{B}$) PLUS 1
L	H	L	H	F = $\overline{B}$	F = AB PLUS (A + $\overline{B}$)	F = AB PLUS (A + $\overline{B}$) PLUS 1
L	H	H	L	F = $\overline{A \odot B}$	F = A MINUS B MINUS 1	F = A MINUS B
L	H	H	H	F = A + $\overline{B}$	F = A + $\overline{B}$	F = (A + $\overline{B}$) PLUS 1
H	L	L	L	F = $\overline{A}$B	F = A PLUS (A + B)	F = A PLUS (A + B) PLUS 1
H	L	L	H	F = A $\odot$ B	F = A PLUS B	F = A PLUS B PLUS 1
H	L	H	L	F = B	F = A$\overline{B}$ PLUS (A + B)	F = A$\overline{B}$ PLUS (A + B) PLUS 1
H	L	H	H	F = A + B	F = (A + B)	F = (A + B) PLUS 1
H	H	L	L	F = 0	F = A PLUS A*	F = A PLUS A PLUS 1
H	H	L	H	F = A$\overline{B}$	F = AB PLUS A	F = AB PLUS A PLUS 1
H	H	H	L	F = AB	F = A$\overline{B}$ PLUS A	F = A$\overline{B}$ PLUS A PLUS 1
H	H	H	H	F = A	F = A	F = A PLUS 1

TABLE 2

SELECTION				ACTIVE-HIGH DATA		
				M = H	M = L; ARITHMETIC OPERATIONS	
S3	S2	S1	S0	LOGIC FUNCTIONS	$\overline{C}_n$ = H (no carry)	$\overline{C}_n$ = L (with carry)
L	L	L	L	F = $\overline{A}$	F = A	F = A PLUS 1
L	L	L	H	F = $\overline{A+B}$	F = A + B	F = (A + B) PLUS 1
L	L	H	L	F = A$\overline{B}$	F = A + $\overline{B}$	F = (A + $\overline{B}$) PLUS 1
L	L	H	H	F = 0	F = MINUS 1 (2's COMPL)	F = ZERO
L	H	L	L	F = $\overline{AB}$	F = A PLUS A$\overline{B}$	F = A PLUS A$\overline{B}$ PLUS 1
L	H	L	H	F = $\overline{B}$	F = (A + B) PLUS A$\overline{B}$	F = (A + B) PLUS A$\overline{B}$ PLUS 1
L	H	H	L	F = A $\odot$ B	F = A MINUS B MINUS 1	F = A MINUS B
L	H	H	H	F = A$\overline{B}$	F = A$\overline{B}$ MINUS 1	F = A$\overline{B}$
H	L	L	L	F = $\overline{A}$ + B	F = A PLUS AB	F = A PLUS AB PLUS 1
H	L	L	H	F = $\overline{A \odot B}$	F = A PLUS B	F = A PLUS B PLUS 1
H	L	H	L	F = B	F = (A + $\overline{B}$) PLUS AB	F = (A + $\overline{B}$) PLUS AB PLUS 1
H	L	H	H	F = AB	F = AB MINUS 1	F = AB
H	H	L	L	F = 1	F = A PLUS A*	F = A PLUS A PLUS 1
H	H	L	H	F = A + $\overline{B}$	F = (A + B) PLUS A	F = (A + B) PLUS A PLUS 1
H	H	H	L	F = A + B	F = (A + $\overline{B}$) PLUS A	F = (A + $\overline{B}$) PLUS A PLUS 1
H	H	H	H	F = A	F = A MINUS 1	F = A

Table des fonctions générées

functional block diagram



ALU 74181, M = 1: Diagramme synoptique des fonctions logiques

Nous pouvons faire la même chose pour les opérations arithmétiques, mais c'est plus complexe car il faut tenir compte de l'étage précédent — ce que fait cette unité — c'est pourquoi l'expression «*Look ahead*» (*lecture par anticipation*) apparaît dans sa description.

Reportons-nous au diagramme synoptique des fonctions ci-dessous. En code d'addition arithmétique: $C_n = 1$, M = 0 et S3 S2 S1 S0 = 1001, on doit générer Ai plus Bi plus Ci−1 et le report Ci (remarquer l'usage de «plus» au lieu de «+» pour éviter toute confusion avec l'opérateur logique «ou»). Selon les notations qui y figurent:

$$Fi = Xi \oplus Yi \oplus Ci-1$$

avec
$$Xi = \overline{Ai\ \overline{Bi}\ S2 + Ai\ Bi\ S3}$$
$$= \overline{Ai\ Bi}$$

et
$$Yi = \overline{Ai + Bi\ S0 + \overline{Bi}\ S1}$$
$$= \overline{Ai + Bi}$$

En nous reportant à la définition de l'addition arithmétique avec le report Ci−1 de l'étage précédent i−1 nous avons:

Ci−1	Bi	Ai	∑i	Ci
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Ce qui nous donne les équations:

$$\sum i = Ai \oplus Bi \oplus Ci-1$$
$$Ci = Ai\ Bi + (Ai \oplus Bi)\ Ci-1$$

Prouvons que le diagramme électrique nous donne les égalités suivantes.

$$Ci = Ai\ Bi + (Ai \oplus Bi)\ Ci-1$$
$$Fi = \sum i$$

Montrons tout d'abord que $Xi \oplus Yi = Ai + Bi$

En effet $Xi + Yi = \overline{Xi}\, Yi + Xi\, \overline{Yi}$

$= Ai\, Bi\ \overline{\overline{Ai} + \overline{Bi}} + \overline{Ai\, Bi}\ (Ai + Bi)$

$= Ai\, Bi\ \overline{Ai}\, \overline{Bi} + (\overline{Ai} + \overline{Bi})(Ai + Bi)$

$= \overline{Ai}\, Bi + AI\, \overline{Bi}$

$= Ai \oplus Bi$

a) $i = 0$

On a: $C0-1 = \overline{\overline{M}\, \overline{Cn}}$

$= 0$

Donc, $F0 = X0 \oplus Y0 \oplus 0$

$= X0 \oplus Y0$

$= A0 \oplus B0$

$= \sum 0$

b) $i = 1$

On a: $C0 = \overline{\overline{X0} + \overline{Y0}}$

$= \overline{\overline{A0\, BO}\ \overline{A0 + B0}}$

$= A0\, B0\, (A0 + B0)$

$= A0\, B0$

$= A0\, B0 + 0 \cdot (A0 \oplus B0)$, d'où:

$C0 = A0\, B0 + C0-1\, (A0 \oplus B0)$

Donc, $F1 = X1 \oplus Y1 \oplus C0$

$= A1 \oplus B1 \oplus C0$

$= \sum 1$

c) i = 2

On a: $C1 = \overline{\overline{X1}\ X0 + X1\ Y0 + Y1}$

$= (\overline{X1} + \overline{X0})(\overline{X1} + Y0)\ \overline{Y1}$

$= \overline{X1}\ \overline{Y1} + \overline{X1}\ \overline{Y1}\ Y0 + \overline{X1}\ \overline{X0}\ \overline{Y1} + \overline{X0}\ \overline{Y1}\ Y0$

$= \overline{X1}\ \overline{Y1} + \overline{X0}\ \overline{Y1}\ Y0$

$= A1\ B1\ (A1 + B1) + A0\ B0\ (A1 + B1)\ (A0 + B0)$

$= A1\ B1 + A0\ B0\ (A1 + B1)$

$= A1\ B1 + A1\ A0\ B0 + A0\ B1\ B0$

$= A1\ (B1 + A0\ B0) + B1\ (A1 + A0\ B0)$

$= A1\ (B1 + A0\ \overline{B1}\ B0) + B1\ (A1 + \overline{A1}\ A0\ B0)$

$= A1\ B1 + (A1\ \overline{B1} + \overline{A1}\ B1)\ A0\ B0$

$= A1\ B1 + (A1 \oplus B1)\ A0\ B0$, d'où:

$C1 = A1\ B1 + (A1 \oplus B1)\ C0$

Donc, $F2 = X2 \oplus Y2 \oplus C1$

$= A1 \oplus B2 \oplus C1$, soit

$F2 = \sum 2$

d) i = 3

$$C2 = \overline{\overline{A2\,B2}\ \overline{A1\,B1}\ \overline{A0\,B0} + \overline{A2\,B2}\ \overline{A1\,B1}\ \overline{A0} + \overline{B0} + \overline{A2\,B2}\ \overline{A1} + \overline{B1} + \overline{A2} + \overline{B2}}$$

$= (A2\ B2 + A1\ B1 + A0\ B0)\ (A2\ B2 + A1\ B1 + A0 + B0)$
$(A2\ B2 + A1 + B1)\ (A2 + B2)$

$= (A2\,B2 + A1\,B1 + A0\,B0)\ (A2\,B2 + A2\,A1 + A1\,B2$
$+ A2\,B1 + B2\,B1)$

$= A2\,B2 + A2\,A1\,B2\,B1 + A2\,A1\,B1 + A1\,B2\,B1 + A2\,A1\,A0\,B0$
$+ A1\ A0\ B2\ B0 + A2\ A0\ B1\ B0 + A0\ B2\ B1\ B0$

$$= A2\,B2 + A2\,A1\,B1 + A1\,B2\,B1 + A2\,A1\,A0\,B0 + A1\,A0\,B2\,B0 + A2\,A0\,B1\,B0 + A0\,B2\,B1\,B0$$

$$= A2\,B2 + A2\,A1\,\overline{B2}\,B1 + A2\,A1\,A0\,\overline{B2}\,B0 + A2\,A0\,\overline{B2}\,B1\,B0 + \overline{A2}\,A1\,B2\,B1 + \overline{A2}\,A1\,A0\,B2\,B0 + \overline{A2}\,A0\,B2\,B1\,B0$$

$$= A2\,B2 + (A1\,B1 + A1\,A0\,B0 + A0\,B1\,B0)\,(A2\,\overline{B2} + \overline{A2}\,B2)$$

$$= A2\,B2 + [A1\,B1 + A0\,B0\,(A1 \oplus B1)]\,(A2 \oplus B2)$$

$$= A2\,B2 + [A1\,B1 + (A1 \oplus B1)\,C0]\,(A2 \oplus B2), \text{ soit:}$$

$$= A2\,B2 + (A2 \oplus B2)\,C1$$

Donc, $F3 = X3 \oplus Y3 \oplus C2$

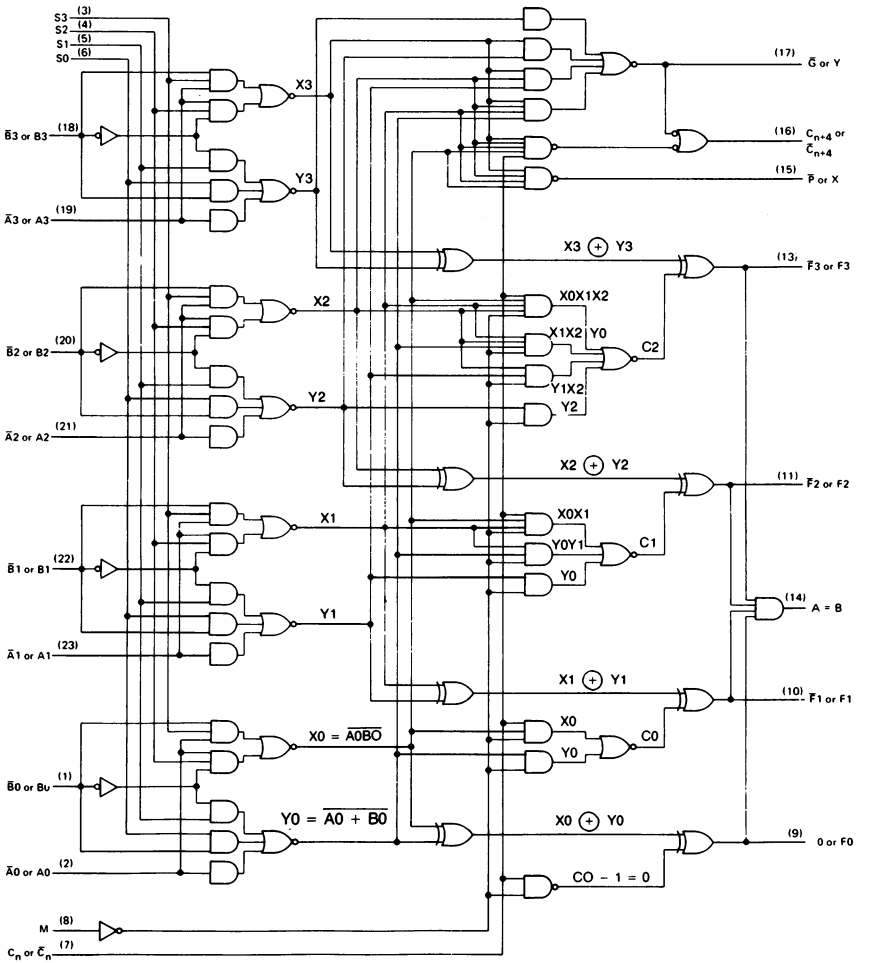
$= A3 \oplus B3 \oplus C2$, soit:

$F3 = \sum 3$

On a bien un additionneur complet de quatre bits.

La notice technique nous apprend qu'on peut aussi faire les opérations avec plusieurs unités en parallèle, donc avec 4, 8, 12, 16, 20, 22, 28, 32 bits. Pour cela il faut acheter un circuit supplémentaire transmettant le report à l'étage suivant: un C1 74182 si on veut un temps d'opération rapide, sinon on peut combiner les circuits directement. (Si $\overline{C_n} = 0$ on a F = A plus B plus 1).

functional block diagram



ALU 74181, M = 0, $\overline{C_n}$ = 1 et S3S2S1S0 = 1001:
diagramme synoptique de l'addition arithmétique.

PROBLÈME 11 — DÉCODEUR DE TROIS LIGNES VERS HUIT LIGNES

Faire la synthèse d'un décodeur octal de trois lignes vers huit lignes.

Solution

Appelons les entrées C, B, A et les sorties Y0, Y1, Y2, Y3, Y4, Y5, Y6, et Y7.

Étape 1 — Mise en équation

Table de vérité

C	B	A	Sorties
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

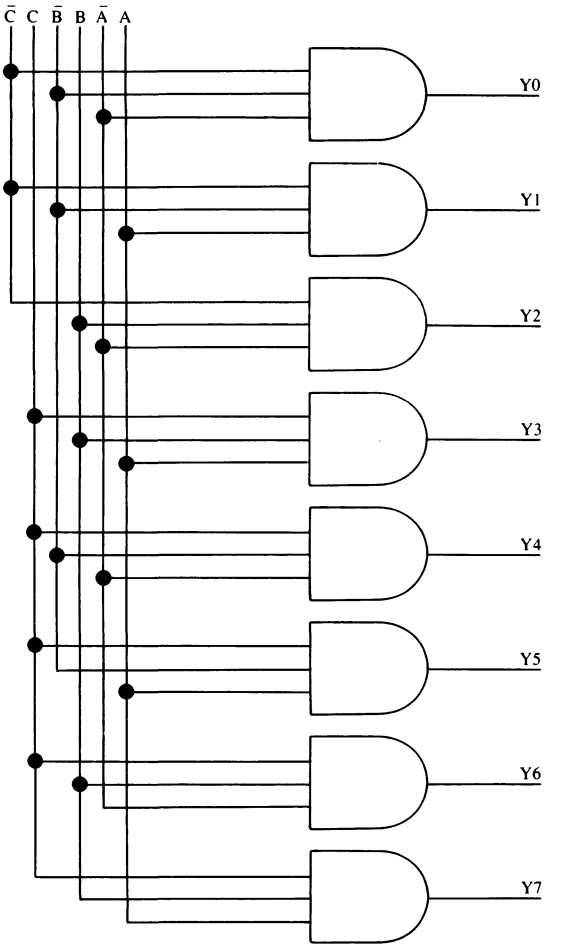
Table de Karnaugh

C \ BA	00	01	11	10
0	Y0	Y1	Y3	Y2
1	Y4	Y5	Y7	Y6

Équations des sorties

$$Y0 = \bar{C}\bar{B}\bar{A}$$
$$Y1 = \bar{C}\bar{B}A$$
$$Y2 = \bar{C}B\bar{A}$$
$$Y3 = \bar{C}BA$$
$$Y4 = C\bar{B}\bar{A}$$
$$Y5 = C\bar{B}A$$
$$Y6 = CB\bar{A}$$
$$Y7 = CBA$$

Étape 2 — Implantation



Décodeur octal de trois lignes vers huit lignes

La synthèse des décodeurs de deux lignes vers quatre lignes et de quatre lignes vers seize lignes repose sur le même principe. Parmi les circuits intégrés d'implantation de ces fonctions, citons:

— le 74155: décodeur de deux lignes vers quatre lignes

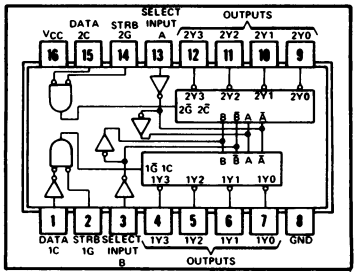
— le 74138: décodeur de trois lignes vers huit lignes

— le 74154: décodeur de quatre lignes vers seize lignes.

Le diagramme de brochage, le diagramme logique et la table de fonctions de chacun d'eux apparaissent ci-dessous.

Le terme *démultiplexeur* utilisé pour désigner ces circuits, précise qu'il existe une entrée supplémentaire pour les données ou *l'échantillonnage* ou pour les deux. Le C1 74155, par exemple, à deux décodeurs/démultiplexeurs de deux lignes vers quatre lignes comporte les deux entrées DATA (broches 1, 15) et STROBE (broches 2, 14).

Nous verrons une application d'un tel circuit, mais nous définirons auparavant un autre type de circuit, le *multiplexeur*.



FUNCTION TABLES
2-LINE-TO-4-LINE DECODER
OR 1-LINE-TO-4-LINE DEMULTIPLEXER

INPUTS				OUTPUTS			
SELECT		STROBE	DATA				
B	A	1G	1C	1Y0	1Y1	1Y2	1Y3
X	X	H	X	H	H	H	H
L	L	L	H	L	H	H	H
L	H	L	H	H	L	H	H
H	L	L	H	H	H	L	H
H	H	L	H	H	H	H	L
X	X	X	L	H	H	H	H

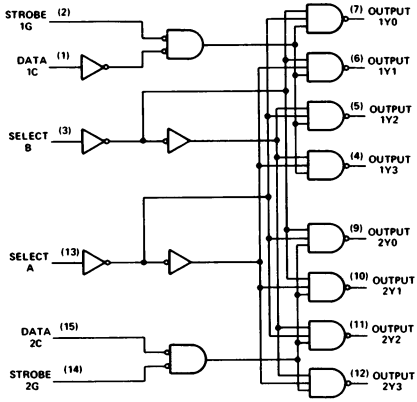
INPUTS				OUTPUTS			
SELECT		STROBE	DATA				
B	A	2G	2C	2Y0	2Y1	2Y2	2Y3
X	X	H	X	H	H	H	H
L	L	L	L	L	H	H	H
L	H	L	L	H	L	H	H
H	L	L	L	H	H	L	H
H	H	L	L	H	H	H	L
X	X	X	H	H	H	H	H

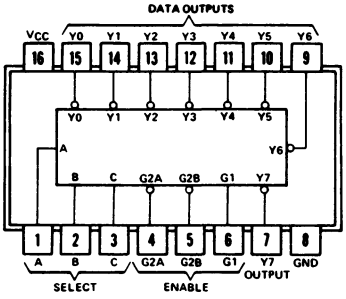
FUNCTION TABLE
3-LINE-TO-8-LINE DECODER
OR 1-LINE-TO-8-LINE DEMULTIPLEXER

INPUTS				OUTPUTS							
SELECT			STROBE OR DATA	(0)	(1)	(2)	(3)	(4)	(5)	(6)	(7)
C†	B	A	G‡	2Y0	2Y1	2Y2	2Y3	1Y0	1Y1	1Y2	1Y3
X	X	X	H	H	H	H	H	H	H	H	H
L	L	L	L	L	H	H	H	H	H	H	H
L	L	H	L	H	L	H	H	H	H	H	H
L	H	L	L	H	H	L	H	H	H	H	H
L	H	H	L	H	H	H	L	H	H	H	H
H	L	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	L	H	H
H	H	L	L	H	H	H	H	H	H	L	H
H	H	H	L	H	H	H	H	H	H	H	L

†C = inputs 1C and 2C connected together
‡G = inputs 1G and 2G connected together

FUNCTIONAL BLOCK DIAGRAM

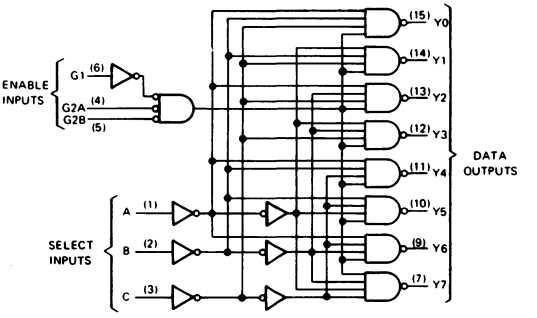




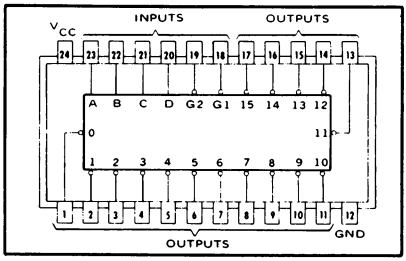
FUNCTION TABLE

INPUTS					OUTPUTS							
ENABLE		SELECT										
G1	G2*	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

*G2 = G2A + G2B



Décodeurs-démultiplexeurs 74 LS 138, 74S138

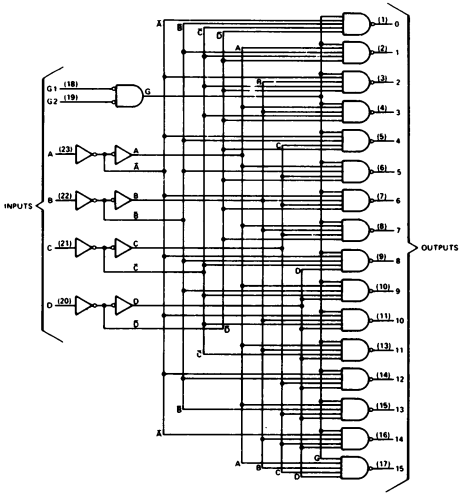


FUNCTION TABLE

INPUTS						OUTPUTS														
G1	G2	D	C	B	A	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
L	L	L	L	L	L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	L	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H
L	L	L	H	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H
L	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H
L	L	H	L	L	L	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H
L	L	H	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H
L	L	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H
L	L	H	H	L	L	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H
L	L	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L
L	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	L	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H

H = high level, L = low level, X = irrelevant

functional block diagram and schematics of inputs and outputs



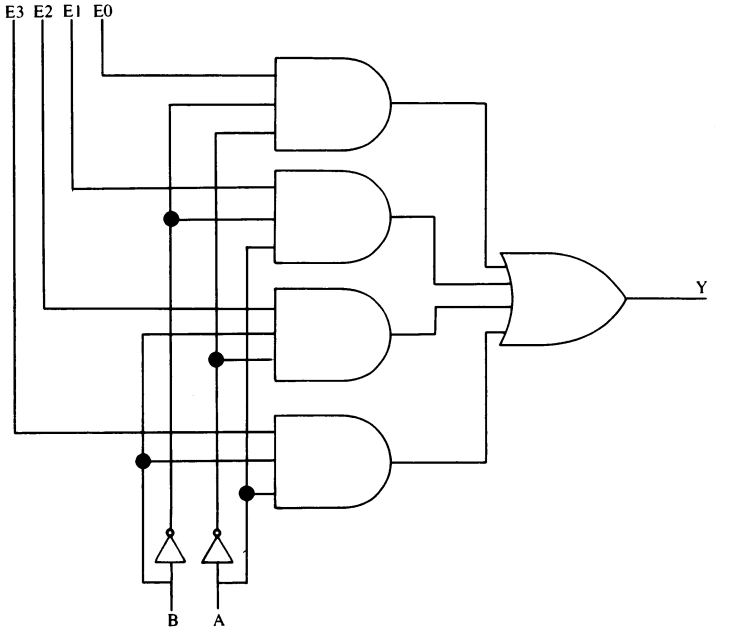
PROBLÈME 12 — MULTIPLEXEUR

Le multiplexeur est un circuit qui permet de sélectionner une ligne d'entrée par une adresse et de faire apparaître à la sortie l'état de cette ligne, c'est-à-dire un niveau H ou L (haut ou bas).

Multiplexeur de quatre lignes: l'adresse aura deux lignes et on veut à la sortie l'état de la ligne E0, E1, E2 et E3, le chiffre représentant l'adresse affichée par les deux lignes B et A. Soit Y la sortie. On veut:

B	A	Y
0	0	E0
0	1	E1
1	0	E2
1	1	E3

D'où l'équation $Y = \bar{A}\bar{B}E0 + A\bar{B}E1 + \bar{A}BE2 + ABE3$ et l'implantation:



La sortie Y reproduira le niveau logique de la ligne de données sélectionnée par l'adresse BA.

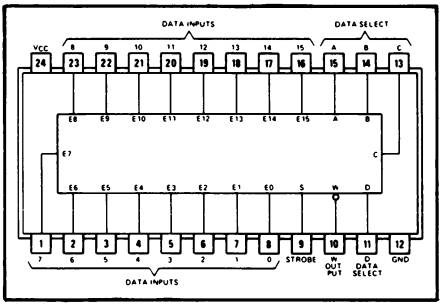
On appelle ce circuit multiplexeur, car il permet de transmettre sur la ligne Y les données apparaissant sur les quatre lignes d'entrée E0 à E3 en faisant varier l'adresse.

Parmi les multiplexeurs disponibles en circuits intégrés, signalons le 74150 à seize entrées, le 74151 à huit entrées et le 74152 à huit entrées.

Le 74151 présente deux sorties, Y et W. La sortie W est le complément de Y.

Le circuit 74150 et le 74152 ont seulement une sortie W qui donne le complément de la valeur de la ligne sélectionnée. le 74150 et le 74151 ont une entrée d'échantillonnage ou de *validation* (Strobe, enable) qui met la sortie W à H si cette entrée est H et la sortie Y, dans le cas du 74151, à L.

Les schémas ci-dessous représentent le schéma de brochage, la table de fonctions et le diagramme logique des circuits intégrés 74150, 74151 et 74152.



FUNCTION TABLE

INPUTS					OUTPUT
SELECT				STROBE	W
D	C	B	A	S	
X	X	X	X	H	H
L	L	L	L	L	$\overline{E0}$
L	L	L	H	L	$\overline{E1}$
L	L	H	L	L	$\overline{E2}$
L	L	H	H	L	$\overline{E3}$
L	H	L	L	L	$\overline{E4}$
L	H	L	H	L	$\overline{E5}$
L	H	H	L	L	$\overline{E6}$
L	H	H	H	L	$\overline{E7}$
H	L	L	L	L	$\overline{E8}$
H	L	L	H	L	$\overline{E9}$
H	L	H	L	L	$\overline{E10}$
H	L	H	H	L	$\overline{E11}$
H	H	L	L	L	$\overline{E12}$
H	H	L	H	L	$\overline{E13}$
H	H	H	L	L	$\overline{E14}$
H	H	H	H	L	$\overline{E15}$

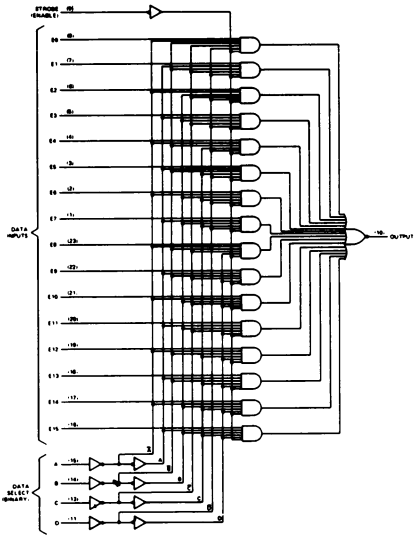
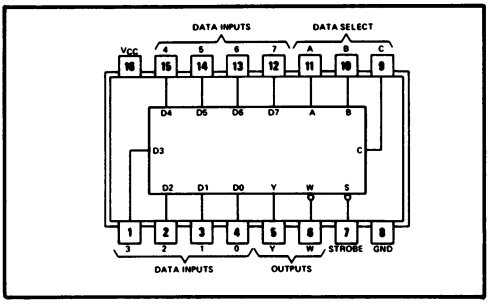


Schéma de brochage,
table des fonctions et implantation des fonctions du CI 74150.



FUNCTION TABLE

INPUTS				OUTPUTS	
SELECT			STROBE	Y	W
C	B	A	S		
X	X	X	H	L	H
L	L	L	L	D0	$\overline{D0}$
L	L	H	L	D1	$\overline{D1}$
L	H	L	L	D2	$\overline{D2}$
L	H	H	L	D3	$\overline{D3}$
H	L	L	L	D4	$\overline{D4}$
H	L	H	L	D5	$\overline{D5}$
H	H	L	L	D6	$\overline{D6}$
H	H	H	L	D7	$\overline{D7}$

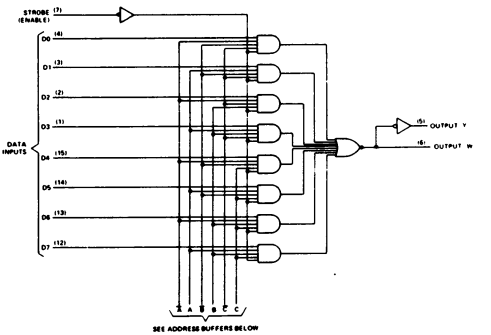
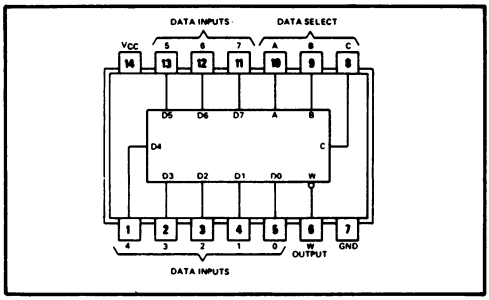


Schéma de brochage,
table de fonctions et implantation des fonctions du C1 74151.



FUNCTION TABLE

SELECT INPUTS			OUTPUT W
C	B	A	
L	L	L	$\overline{D0}$
L	L	H	$\overline{D1}$
L	H	L	$\overline{D2}$
L	H	H	$\overline{D3}$
H	L	L	$\overline{D4}$
H	L	H	$\overline{D5}$
H	H	L	$\overline{D6}$
H	H	H	$\overline{D7}$

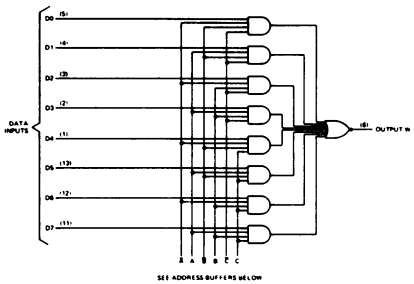
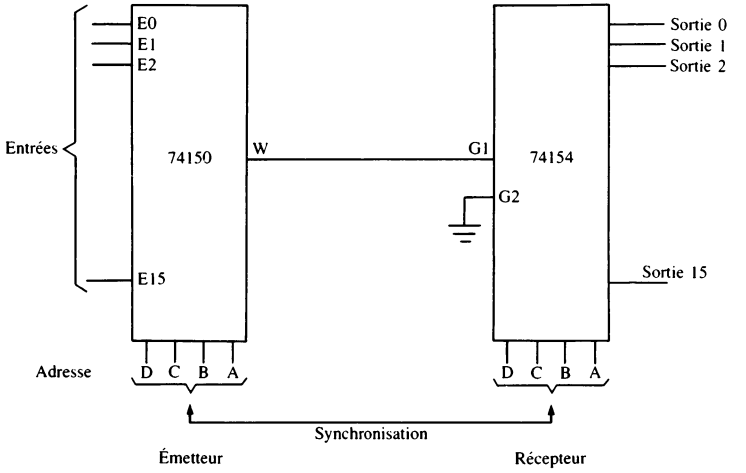


Schéma de brochage,
table de fonctions et implantation des fonctions du C1 74152.

Tampons d'adresse pour les CI 74151 et 74152.

PROBLÈME 13 — TRANSMISSION DE DONNÉES EN BINAIRE

Exemple d'utilisation d'un multiplexeur et d'un démultiplexeur pour la transmission de données binaires.



Si on synchronise les deux adresses, c'est-à-dire si ces deux adresses sont les mêmes en même temps, par exemple 12, alors:

$$W = \overline{E12} \quad \text{et} \quad G1 = W = \overline{E12}$$

si E12 = 1, alors G1 = 0 = L et Sortie 12 = L

si E12 = 0, alors G1 = 1 = H et Sortie 12 = H

d'où Sortie 12 = E12 au moment où l'adresse est 12
= H dans les autres cas (se reporter au manuel de la société Texas Instruments Incorporated).

On peut donc lire la valeur de seize lignes en un temps donné (temps de balayage de toutes les adresses) et la transmettre sur une autre ligne. En synchronisant la lecture du démultiplexeur, on peut reconstituer la valeur binaire de la ligne lue par l'adresse et la transmettre sur seize lignes.

PROBLÈME 14 — DÉCODEURS DE BCD, BCD PLUS TROIS, GRAY PLUS TROIS, VERS DIX LIGNES

Effectuer le décodage de BCD vers dix lignes seulement, les deux autres sont laissés en exercices.

Solution

Étape 1 — Mise en équation

DC \ BA	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	X	X	X	X
10	8	9	X	X

Les X indiquent des conditions d'entrée qui ne peuvent survenir. Les équations du circuit sont:

$0 = \bar{D}\bar{C}\bar{B}\bar{A}$ $\quad 4 = \bar{D}C\bar{B}\bar{A}$ $\quad 8 = D\bar{C}\bar{B}\bar{A}$

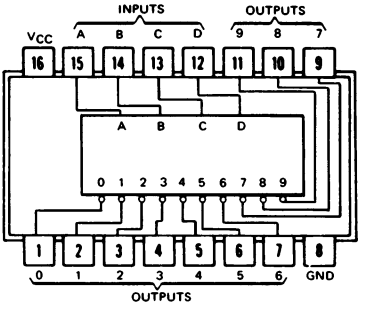
$1 = \bar{D}\bar{C}\bar{B}A$ $\quad 5 = \bar{D}C\bar{B}A$ $\quad 9 = D\bar{C}\bar{B}A$

$2 = \bar{D}\bar{C}B\bar{A}$ $\quad 6 = \bar{D}CB\bar{A}$

$3 = \bar{D}\bar{C}BA$ $\quad 7 = \bar{D}CBA$

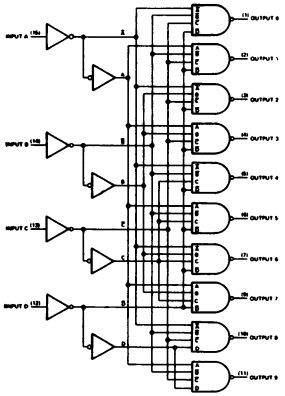
Étape 2 — Implantation

Elle est réalisée par le circuit 7442. On trouvera ci-dessous le schéma de brochage du C1 7442 (de BCD à décimal) 7443 (de BCD plus trois à décimal) et 7444 (de Gray plus trois à décimal) ainsi que leur table de fonctions et leur diagramme logique.

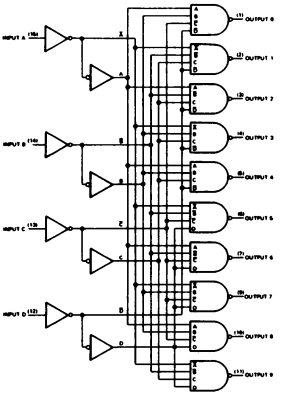


FUNCTION TABLE

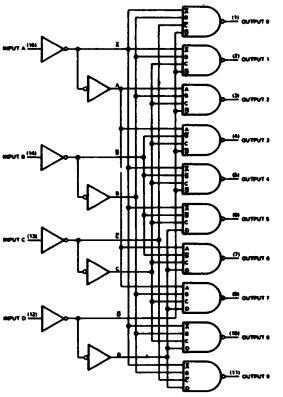
NO.	'42A, 'L42, 'LS42 BCD INPUT				'43A, 'L43 EXCESS-3-INPUT				'44A, 'L44 EXCESS-3-GRAY INPUT				ALL TYPES DECIMAL OUTPUT									
	D	C	B	A	D	C	B	A	D	C	B	A	0	1	2	3	4	5	6	7	8	9
0	L	L	L	L	L	L	H	H	L	L	H	L	L	H	H	H	H	H	H	H	H	H
1	L	L	L	H	L	H	L	L	L	H	H	L	H	L	H	H	H	H	H	H	H	H
2	L	L	H	L	L	H	L	H	L	H	H	H	H	H	L	H	H	H	H	H	H	H
3	L	L	H	H	L	H	H	L	L	H	L	H	H	H	H	L	H	H	H	H	H	H
4	L	H	L	L	L	H	H	H	L	H	L	L	H	H	H	H	L	H	H	H	H	H
5	L	H	L	H	H	L	L	L	H	H	L	L	H	H	H	H	H	L	H	H	H	H
6	L	H	H	L	H	L	L	H	H	H	L	H	H	H	H	H	H	H	L	H	H	H
7	L	H	H	H	H	L	H	L	H	H	H	H	H	H	H	H	H	H	H	L	H	H
8	H	L	L	L	H	L	H	H	H	H	H	L	H	H	H	H	H	H	H	H	L	H
9	H	L	L	H	H	H	L	L	H	L	H	L	H	H	H	H	H	H	H	H	H	L
INVALID	H	L	H	L	H	H	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H
	H	L	H	H	H	H	H	L	H	L	L	H	H	H	H	H	H	H	H	H	H	H
	H	H	L	L	H	H	H	H	H	L	L	L	H	H	H	H	H	H	H	H	H	H
	H	H	L	H	L	L	L	L	L	L	L	L	H	H	H	H	H	H	H	H	H	H
	H	H	H	L	L	L	L	H	L	L	L	H	H	H	H	H	H	H	H	H	H	H
	H	H	H	H	L	L	H	L	L	L	H	H	H	H	H	H	H	H	H	H	H	H



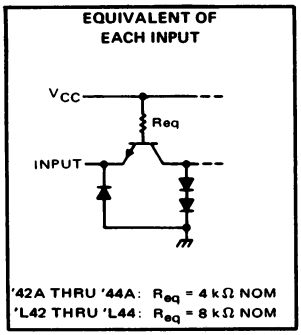
'42A, 'L42, 'LS42
BCD-TO-DECIMAL DECODERS

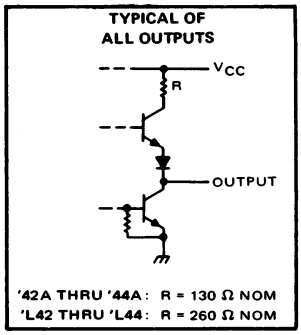


'43A, 'L43
EXCESS-3-TO-DECIMAL DECODERS



'44A, 'L44
EXCESS-3-GRAY-TO-DECIMAL DECODERS





C1 décodeurs 7442, 7443, 7444 types A et L vers dix lignes

Si un code à l'entrée d'un décodeur est incorrect, 1100 par exemple, pour le décodeur BCD, aucun des chiffres ne s'allume. C'est un cas de réjection de données incorrectes.

Comme on a une table de Karnaugh incomplète, on peut combiner les cases avec des X pour obtenir le circuit le plus simple possible. On a alors:

$0 = \bar{D}\bar{C}\bar{B}\bar{A}$ $\quad$ $5 = C\bar{B}A$

$1 = \bar{D}\bar{C}\bar{B}A$ $\quad$ $6 = CB\bar{A}$

$2 = \bar{C}B\bar{A}$ $\quad$ $7 = CBA$

$3 = \bar{C}BA$ $\quad$ $8 = D\bar{A}$

$4 = C\bar{B}\bar{A}$ $\quad$ $9 = DA$

On obtient un circuit beaucoup plus simple mais on n'aura plus réjection de données incorrectes.

Pour DCBA = 1100, par exemple, les chiffres 4 et 8 s'allumeront en même temps.

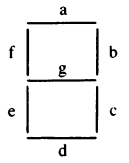
À titre d'exercice, déterminer les chiffres qui s'allumeront pour les entrées

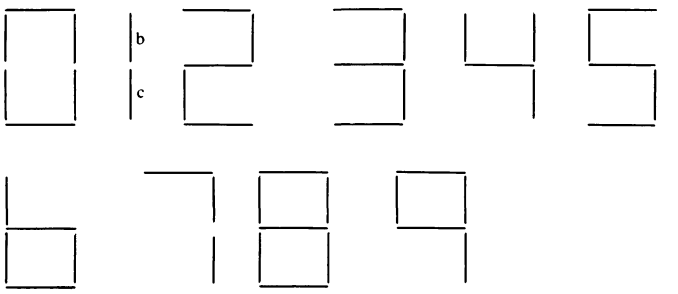
D	C	B	A	
1	0	1	0	:
1	0	1	1	:
1	1	0	0	:
1	1	0	1	:
1	1	1	0	:
1	1	1	1	:

PROBLÈME 15 — DÉCODEUR DE BCD VERS SEPT LIGNES

Solution

Les sept segments lumineux, disposés de la façon suivante, permettent l'écriture de tous les chiffres et aussi d'autres symboles.

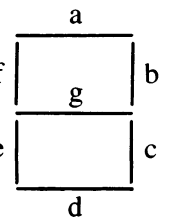




Étape 1 — Table de fonctions

Pour analyser le décodeur, dressons la table des sorties en fonction des entrées. Les lignes, numérotées de 0 à 9, sont les entrées du nombre BCD à décoder. Les colonnes indiquent les segments, 1 est attribué au segment allumé et 0, au segment éteint. On obtient la table ci-après.

	a	b	c	d	e	f	g
0	1	1	1	1	1	1	0
1	0	1	1	0	0	0	0
2	1	1	0	1	1	0	1
3	1	1	1	1	0	0	1
4	0	1	1	0	0	1	1
5	1	0	1	1	0	1	1
6	0	0	1	1	1	1	1
7	1	1	1	0	0	0	0
8	1	1	1	1	1	1	1
9	1	1	1	0	0	1	1



On voit que pour afficher 5, par exemple, il faut éteindre les segments b et e.

Étape 2 — Mise en équation

Pour obtenir les équations de ce décodeur, il faut établir la table de Karnaugh de chaque segment. On aura donc sept tables de Karnaugh.

Ces tables figurent ci-dessous. Les 0 étant moins nombreux, l'écriture des équations de commande d'extinction des segments, laissée au lecteur, sera plus simple. Considérer les X pertinents.

Segment a

DC \ BA	00	01	11	10
00	1	0	1	1
01	0	1	1	0
11	x	x	x	x
10	1	1	x	x

$\bar{a} = A \cdot \bar{B} \cdot \bar{C} \cdot \bar{D} + \bar{A} \cdot C$

Segment b

DC \ BA	00	01	11	10
00	1	1	1	1
01	1	0	1	0
11	x	x	x	x
10	1	1	x	x

$\bar{b} = A \cdot \bar{B} \cdot C + \bar{A} \cdot B \cdot C$

Segment c

DC \ BA	00	01	11	10
00	1	1	1	0
01	1	1	1	1
11	x	x	x	x
10	1	1	x	x

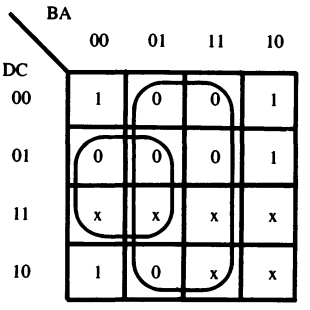
$\bar{c} = \bar{A} \cdot B \cdot \bar{C}$

Segment d

DC \ BA	00	01	11	10
00	1	0	1	1
01	0	1	0	1
11	x	x	x	x
10	1	0	x	x

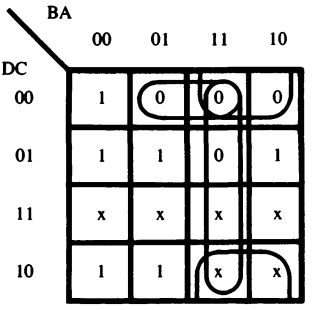
$\bar{d} = \bar{A} \cdot \bar{B} \cdot C + A \cdot B \cdot C + A \cdot \bar{B} \cdot \bar{C}$

Segment e



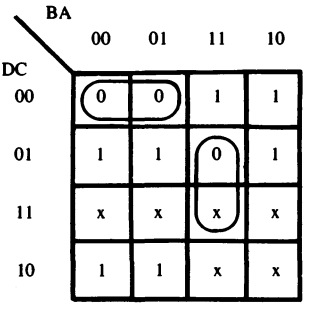
$\bar{e} = A + \bar{B} \cdot C$

Segment f



$\bar{f} = A \cdot \bar{C} \cdot \bar{D} + A \cdot B + B \cdot \bar{C}$

Segment g

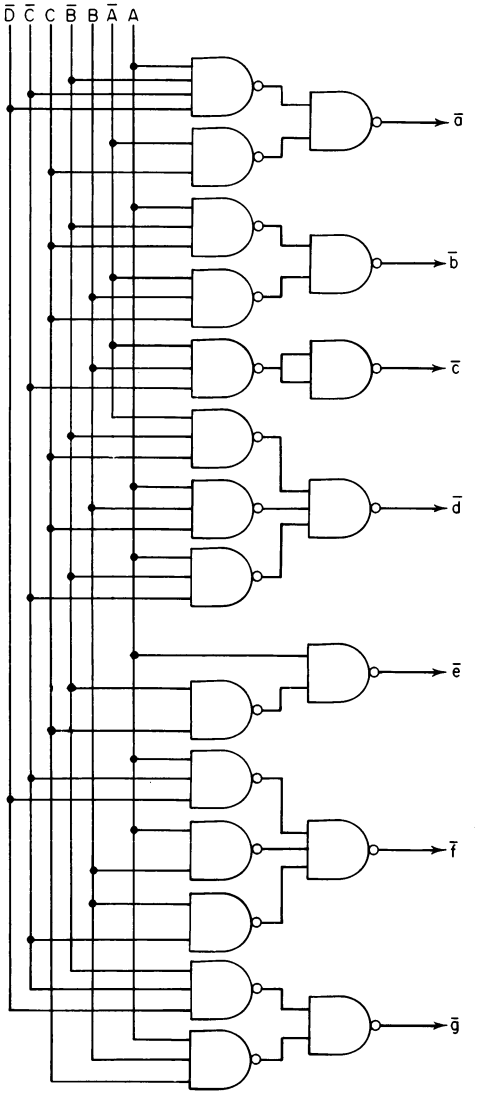


$\bar{g} = \bar{B} \cdot \bar{C} \cdot \bar{D} + A \cdot B \cdot C$

Tables de Karnaugh d'un décodeur à 7 segment.

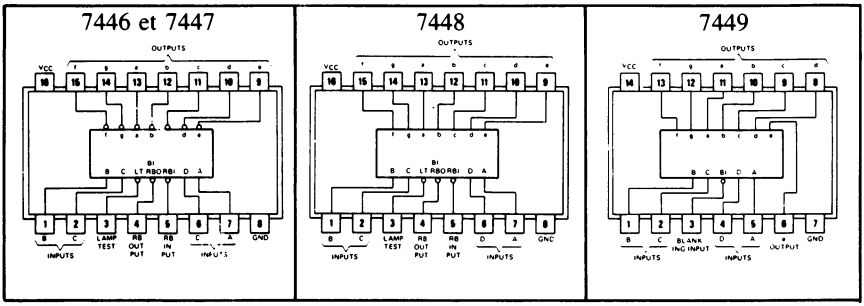
Étape 3 — Implantation

Le circuit logique correspondant à ces équations est donné ci-dessous. Il n'y a pas réjection des données incorrectes (DCBA supérieur à 1001) puisque les X des tables de Karnaugh ont été considérés.

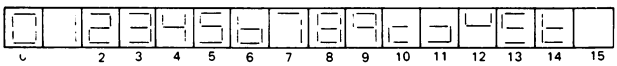


Comme exemple de décodeur de BCD à sept segments, citons les circuits intégrés 7446, 7447, 7448 et 7449. Le schéma de brochage, la table de fonctions ainsi que le diagramme logique de chacun de ces CI sont donnés ci-après.

Parmi les caractéristiques techniques de ces décodeurs, relevons l'affichage de signes particuliers en plus des nombres décimaux. Voir les tables de fonctions et implantations ci-dessous et le manuel de la société Texas Instruments pour toute description supplémentaire.



SEGMENT IDENTIFICATION



NUMERICAL DESIGNATIONS AND RESULTANT DISPLAYS

7446 et 7447

'46A, '47A, 'L46, 'L47, 'LS47 FUNCTION TABLE

DECIMAL OR FUNCTION	INPUTS						BI/RBO†	OUTPUTS							NOTE
	LT	RBI	D	C	B	A		a	b	c	d	e	f	g	
0	H	H	L	L	L	L	H	ON	ON	ON	ON	ON	ON	OFF	1
1	H	X	L	L	L	H	H	OFF	ON	ON	OFF	OFF	OFF	OFF	
2	H	X	L	L	H	L	H	ON	ON	OFF	ON	ON	OFF	ON	
3	H	X	L	L	H	H	H	ON	ON	ON	ON	OFF	OFF	ON	
4	H	X	L	H	L	L	H	OFF	ON	ON	OFF	OFF	ON	ON	
5	H	X	L	H	L	H	H	ON	OFF	ON	ON	OFF	ON	ON	
6	H	X	L	H	H	L	H	OFF	OFF	ON	ON	ON	ON	ON	
7	H	X	L	H	H	H	H	ON	ON	ON	OFF	OFF	OFF	OFF	
8	H	X	H	L	L	L	H	ON	ON	ON	ON	ON	ON	ON	
9	H	X	H	L	L	H	H	ON	ON	ON	OFF	OFF	ON	ON	
10	H	X	H	L	H	L	H	OFF	OFF	OFF	ON	ON	OFF	ON	
11	H	X	H	L	H	H	H	OFF	OFF	ON	ON	OFF	OFF	ON	
12	H	X	H	H	L	L	H	OFF	ON	OFF	OFF	OFF	ON	ON	
13	H	X	H	H	L	H	H	ON	OFF	OFF	ON	OFF	ON	ON	
14	H	X	H	H	H	L	H	OFF	OFF	OFF	ON	ON	ON	ON	
15	H	X	H	H	H	H	H	OFF	OFF	OFF	OFF	OFF	OFF	OFF	
BI	X	X	X	X	X	X	L	OFF	OFF	OFF	OFF	OFF	OFF	OFF	2
RBI	H	L	L	L	L	L	L	OFF	OFF	OFF	OFF	OFF	OFF	OFF	3
LT	L	X	X	X	X	X	H	ON	ON	ON	ON	ON	ON	ON	4

H = high level, L = low level, X = irrelevant

NOTES: 1. The blanking input (BI) must be open or held at a high logic level when output functions 0 through 15 are desired. The ripple blanking input (RBI) must be open or high if blanking of a decimal zero is not desired.

2. When a low logic level is applied directly to the blanking input (BI), all segment outputs are off regardless of the level of any other input.

3. When ripple-blanking input (RBI) and inputs A, B, C, and D are at a low level with the lamp test input high, all segment outputs go off and the ripple blanking output (RBO) goes to a low level (response condition).

7448

FUNCTION TABLE

DECIMAL OR FUNCTION	INPUTS						BI/RBO†	OUTPUTS							NOTE
	LT	RBI	D	C	B	A		a	b	c	d	e	f	g	
0	H	H	L	L	L	L	H	H	H	H	H	H	H	L	1
1	H	X	L	L	L	H	H	L	H	H	L	L	L	L	
2	H	X	L	L	H	L	H	H	H	L	H	H	L	H	
3	H	X	L	L	H	H	H	H	H	H	H	L	L	H	
4	H	X	L	H	L	L	H	L	H	H	L	L	H	H	
5	H	X	L	H	L	H	H	H	L	H	H	L	H	H	
6	H	X	L	H	H	L	H	L	L	H	H	H	H	H	
7	H	X	L	H	H	H	H	H	H	H	L	L	L	L	
8	H	X	H	L	L	L	H	H	H	H	H	H	H	H	
9	H	X	H	L	L	H	H	H	H	H	L	L	H	H	
10	H	X	H	L	H	L	H	L	L	L	H	H	L	H	
11	H	X	H	L	H	H	H	L	L	H	H	L	L	H	
12	H	X	H	H	L	L	H	L	H	L	L	L	H	H	
13	H	X	H	H	L	H	H	H	L	L	H	L	H	H	
14	H	X	H	H	H	L	H	L	L	L	H	H	H	H	
15	H	X	H	H	H	H	H	L	L	L	L	L	L	L	
BI	X	X	X	X	X	X	L	L	L	L	L	L	L	L	2
RBI	H	L	L	L	L	L	L	L	L	L	L	L	L	L	3
LT	L	X	X	X	X	X	H	H	H	H	H	H	H	H	4

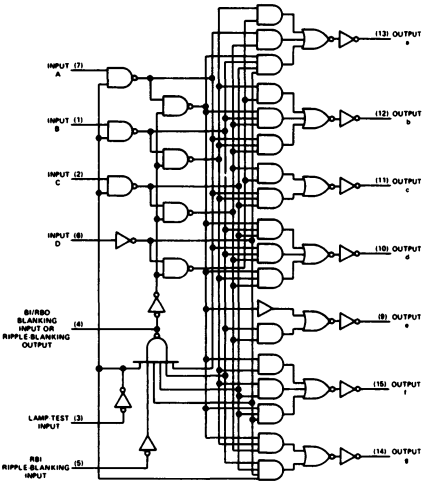
7449

FUNCTION TABLE

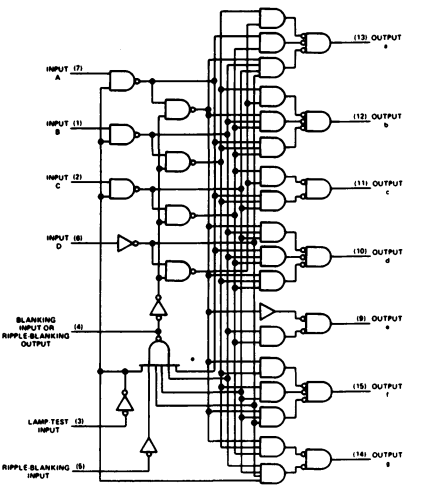
DECIMAL OR FUNCTION	INPUTS					OUTPUTS							NOTE
	D	C	B	A	BI	a	b	c	d	e	f	g	
0	L	L	L	L	H	H	H	H	H	H	H	L	1
1	L	L	L	H	H	L	H	H	L	L	L	L	
2	L	L	H	L	H	H	H	L	H	H	L	H	
3	L	L	H	H	H	H	H	H	H	L	L	H	
4	L	H	L	L	H	L	H	H	L	L	H	H	
5	L	H	L	H	H	H	L	H	H	L	H	H	
6	L	H	H	L	H	L	L	H	H	H	H	H	
7	L	H	H	H	H	H	H	H	L	L	L	L	
8	H	L	L	L	H	H	H	H	H	H	H	H	
9	H	L	L	H	H	H	H	H	L	L	H	H	
10	H	L	H	L	H	L	L	L	H	H	L	H	
11	H	L	H	H	H	L	L	H	H	L	L	H	
12	H	H	L	L	H	L	H	L	L	L	H	H	
13	H	H	L	H	H	H	L	L	H	L	H	H	
14	H	H	H	L	H	L	L	L	H	H	H	H	
15	H	H	H	H	H	L	L	L	L	L	L	L	
BI	X	X	X	X	L	L	L	L	L	L	L	L	2

functional block diagrams

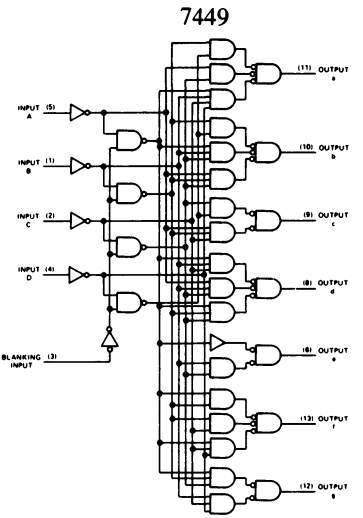
7446 et 7447



7448



7449



Nous venons d'étudier un certain nombre d'applications de la logique combinatoire. Cela nous a permis, en même temps, de passer en revue certains circuits disponibles sur le marché. Les catalogues des fabricants de circuits intégrés offrent d'autres technologies d'implantation des fonctions logiques (CMOS, ECL, etc.) et d'autres fonctions logiques implantées par des circuits intégrés SS1 *(Small Scale Integration)* MSI *(Medium Scale Integration)* et LSI *(Large Scale Integration).* Tous ces points seront approfondis dans le cours de techniques numériques consacré à l'étude de l'utilisation dans des systèmes plus complexes de toutes les fonctions étudiées dans le présent volume.

5

Bascules, compteurs, registres à décalage

5-1 OBJECTIFS

Savoir analyser le fonctionnement et utiliser des composants à mémoire comme les bascules, les compteurs ou les registres à décalage.

1. Être capable d'expliquer la différence entre un circuit combinatoire et une circuit séquentiel.
2. Savoir analyser le fonctionnement d'une bascule.
3. Savoir utiliser les différentes bascules à circuits intégrés.
4. Comprendre le fonctionnement et concevoir des circuits simples utilisant une ou plusieurs bascules.
5. Savoir construire des compteurs et des registres à décalage à l'aide de bascules.
6. Énumérer les avantages et les inconvénients des circuits synchrones et asynchrones.
7. Savoir utiliser des circuits intégrés à moyenne ou grande échelle pour implanter des compteurs et des registres à décalage.

5-2 INTRODUCTION

Pour illustrer ce qu'on entend par logique séquentielle, étudions le problème suivant:

Problème Construire un circuit qui conserve une valeur binaire, 0 ou 1, une fois que cette valeur lui a été précisée.

Soit un circuit à deux entrées $\overline{S}$, $\overline{R}$ et une sortie Q. Si $\overline{S} = 0$ et $\overline{R} = 1$, on veut que $Q = 1$ et conserve cette valeur lorsqu'on passe à $\overline{S} = \overline{R} = 1$ ($\overline{S} = \overline{R} = 1$ peuvent correspondre à des broches en l'air en technologie TTL).

De la même façon, si $\overline{S} = 1$ et $\overline{R} = 0$, on veut que $Q = 0$ et conserve cette valeur lorsqu'on passe à $\overline{S} = \overline{R} = 1$.

Solution Dressons la table de vérité de ce circuit.

$\bar{S}$	$\bar{R}$	Q
0	0	$\not{0}$
0	1	1
1	0	0
1	1	(1) (0)

a) pour $\bar{S} = \bar{R} = 0$, Q n'est pas déterminé par l'énoncé du problème, donc $\not{0}$ dans la table.

b) pour $\bar{S} = \bar{R} = 1$, Q = 1 ou 0: pas de *changement d'état*, selon la valeur antérieure de $\bar{S}$; car l'énoncé du problème exige que Q conserve la valeur commandée.

En ignorant la première ligne, ce problème peut également s'énoncer: Q = 1 pour $\bar{S} = \bar{R} = 1$ si on part de la ligne 2 et Q = 0 pour $\bar{S} = \bar{R} = 1$ si on part de la ligne 3.

Nous avons affaire ici à un problème de *logique séquentielle*: la *valeur actuelle* de la variable de sortie Q ne dépend pas seulement de la valeur actuelle des variables d'entrée $\bar{S}$ et $\bar{R}$ mais aussi de leur *valeur précédente*, 0 ou 1, commandée par les lignes 2 ou 3 de la table de vérité.

Dressons la table de vérité de la façon suivante: plaçons, colonne par colonne, les combinaisons à l'entrée dans l'ordre binaire réfléchi et consacrons une ligne à chaque modification d'une variable d'entrée.

Partons, par exemple de l'*état initial* $\bar{S} = \bar{R} = 1$ et Q = 1. Nous aurons la représentation ci-dessous (figure 5-1).

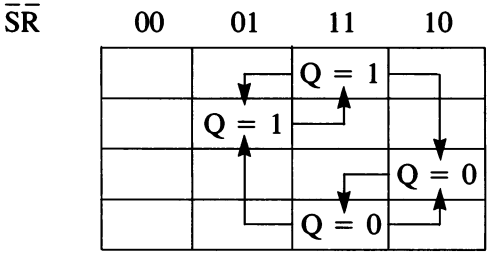


Figure 5-1

La première *séquence* consiste, à partir de l'état initial $\bar{S} = \bar{R} = 1$ et Q = 1, de commander $\bar{S} = 1$ et $\bar{R} = 0$. Sous cette commande, la variable de sortie passe à 0 après un certain retard dû à la propagation de l'ordre d'entrée dans les circuits. Les flèches illustrent le passage d'un état à un

*NdC: S est mis pour Set (*mise à* 1) et R pour Reset (*remise à* 0).

autre, leur angle droit illustre l'existence d'un *état transitoire*. Cette flèche à angle droit précise: a) qu'à partir de l'état initial $\overline{S} = \overline{R} = 1$ et $Q = 1$, on commande $\overline{S} = 1$ et $\overline{R} = 0$ et b) que Q reste à 1 durant un bref instant (état transitoire) et passe ensuite à l'état *stable* 0.

Les différentes flèches de la figure 5-1 indiquent les différents chemins pour passer d'un état à un autre ou de rester dans le même.

Un circuit de ce genre est appelé *bascule*, *verrou* ou *cellule de mémoire* $\overline{S}\,\overline{R}$ car il conserve l'état commandé.

Numérotons maintenant chacun des états et portons la valeur de la sortie à droite de la table. Voir la figure 5-2 ci-dessous.

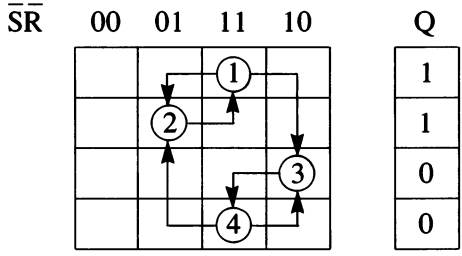


Figure 5-2

Supprimons les flèches et notons la case de l'état transitoire (celle du coude de la flèche) par un numéro égal à celui de l'état stable correspondant. Le numéro de l'état transitoire n'est pas encerclé tandis que celui de l'état stable correspondant l'est. Voir la figure 5-3.

$\overline{S}\overline{R}$	00	01	11	10	Q
		2	(1)	3	1
		(2)	1		1
			4	(3)	0
		2	(4)	3	0

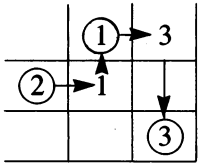
Figure 5-3

La caractéristique de cette table est de comporter un état stable par ligne et un seul.

Les cases vides correspondent à des impossibilités technologiques ou à des entrées non valides.

La première colonne $\overline{S} = \overline{R} = 0$ est vide: on supposera qu'on ne peut effectuer cette commande.

On ne peut pas passer, par exemple, de l'état 2 stable ($\bar{S} = 0$ et $\bar{R} = 1$) à l'état stable ($\bar{S} = 1$ et $\bar{R} = 0$), et réciproquement, d'un seul coup. On transitera par $\bar{S} = \bar{R} = 1$, car on ne peut exiger la simultanéité de deux changements dans un tel circuit soumis à des retards non contrôlables. On effectuera donc le trajet illustré ci-dessous.



La table de la figure 5-3, résultat d'une analyse détaillée du problème, s'appelle la *table primitive des états*.

Mis sous cette forme, ce problème ressemble à un problème de logique combinatoire auquel on a ajouté aux variables *primaires* d'entrée $\bar{S}$ et $\bar{R}$ des variables *secondaires* d'entrée correspondant aux lignes de la table de Karnaugh. Le nombre de ces variables secondaires d'entrée est donc égal à la moitié du nombre de lignes de la table primitive.

On peut, en général, diminuer le nombre de variables secondaires d'entrée. Pour cela on diminue le nombre de lignes de la table selon la règle suivante: a) superposer deux lignes si cela n'entraîne pas l'écriture de deux numéros différents dans une case et b) dans un tel cas, écrire le numéro encerclé de l'état stable, sinon écrire le numéro non encerclé de l'état transitoire.

Dans notre cas, voir la figure 5-3, les lignes 1 et 2 se superposent ainsi que les lignes 3 et 4. On obtient alors la *table* dite *réduite* de la figure 5-4 ci-dessous.

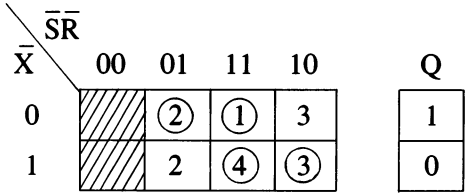


Figure 5-4

$\bar{X}$ désigne la variable secondaire d'entrée dont nous avons besoin. On voit que $X = Q$.

Mise en équation

Remplaçons chaque numéro de case par la valeur de la sortie à l'état stable (voir la table de la figure 5-3).

a) Ajoutons des 1 dans la colonne 1 et considérons les 1 de la table de Karnaugh ci-après.

$\bar{X}$ \ $\bar{S}\bar{R}$	00	01	11	10	
0	1	1	1	0	Q
1	1	1	0	0	

$$\overset{+}{Q} = S + X\bar{R}$$

$$= S + Q\bar{R} \qquad \text{car } X = Q$$

$$= \overline{\overline{S + Q\bar{R}}}$$

$$= \overline{\bar{S} \cdot \overline{Q\bar{R}}}$$

Implantation

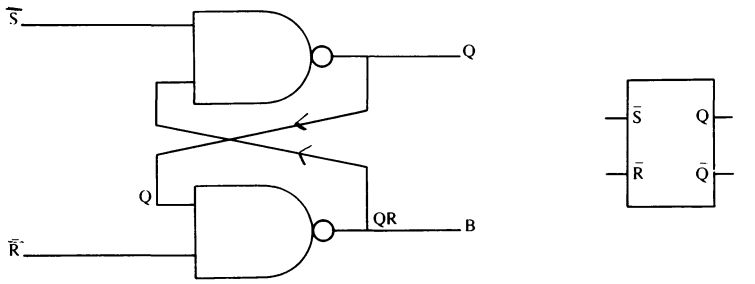


Figure 5-5 Bascule $\bar{S}\bar{R}$ à portes NON-ET

REMARQUE La sortie B est souvent appelée $\bar{Q}$ car si $\bar{R} = 1$ alors $B = Q\bar{R} = \bar{Q}$ et si $\bar{R} = 0$ alors $B = 1$ mais $Q = 0$, donc $B = \bar{Q}$.

La table de vérité ci-dessous résume le fonctionnement de la bascule $\bar{S}\bar{R}$.

$\bar{S}$	$\bar{R}$	Q	$\bar{Q}$	
0	0	1	1	→ *indéfini, interdit* car on ne connaît pas l'état stable après la commande
0	1	1	0	
1	0	0	1	
1	1	Q	$\bar{Q}$	→ déterminé par l'*état* commandé *antérieur*. Pas de changement

b) Ajoutons des 0 dans la colonne 1 et considérons les 0 de la table de Karnaugh ci-dessous.

$\bar{X}$ \ $\bar{S}\bar{R}$	00	01	11	10
0	0	1	1	0
1	0	1	0	0

Q

$$
\begin{aligned}
Q &= \bar{R}\,(S + \bar{X}) \\
&= \bar{R}\,(S + Q) \qquad \text{car } \bar{X} = Q \\
&= \overline{\overline{\bar{R}\,(S + Q)}} \\
&= \overline{R + \overline{S + Q}}
\end{aligned}
$$

Implantation

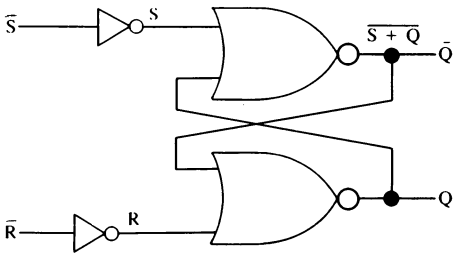


Figure 5-6 Bascule $\bar{S}\bar{R}$ à portes NON-OU.

La table de vérité ci-dessous résume son fonctionnement.

$\bar{S}$	$\bar{R}$	Q	$\bar{Q}$	
0	0	1	1	combinaison interdite
0	1	1	0	
1	0	0	1	
1	1	Q	$\bar{Q}$	pas de changement

On peut faire la même remarque que précédemment pour Q et $\bar{Q}$.

Nous avons montré à l'aide d'un exemple ce qu'est un circuit séquentiel et nous avons ébauché une méthode de résolution de tels circuits.

5-3 CIRCUITS SYNCHRONES ET CIRCUITS ASYNCHRONES

Le type de circuit que nous venons de voir s'appelle un *circuit asynchrone*. Il change d'état lorsque les variables primaires d'entrée changent d'état.

Dans un autre type de circuit, dit *synchrone*, le changement d'état, commandé par les variables primaires d'entrée, est soumis à l'accord d'une commande supplémentaire généralement appelée *signal d'horloge*. On obtient un tel circuit en ajoutant à la bascule $\overline{S}\overline{R}$ à portes NON-ET, par exemple, deux portes NON-ET et un signal d'horloge T. Cela donnera la *bascule SRT* illustrée à la figure 5-7 ci-dessous.

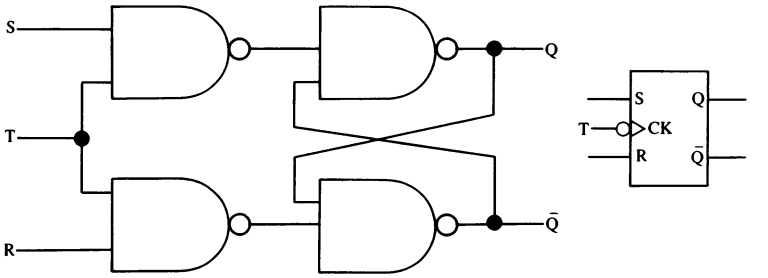


Figure 5-7 Bascule SRT

Cette entrée d'horloge T est considérée dans la synthèse des circuits de ce type comme une variable primaire d'entrée.

Dans le cas de la figure 5-7, le changement d'état est autorisé lorsque le signal d'horloge vaut 1.

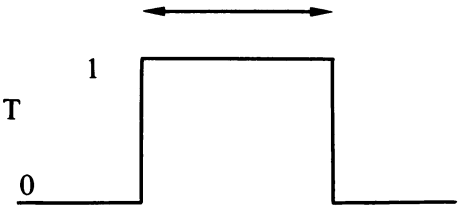
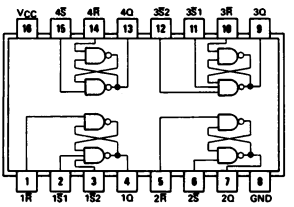


Figure 5-8 Changement d'état autorisé par la valeur 1 de l'horloge

Parmi les circuits intégrés à bascules $\overline{S}\overline{R}$, citons le 74279 dont la table de fonction et le schéma de brochage apparaissent à la figure 5-9.

FUNCTION TABLE

INPUTS		OUTPUT
$\overline{S}$†	$\overline{R}$	Q
H	H	Q_0
L	H	H
H	L	L
L	L	H*



*Niveau de sortie pseudo-stable, peut ne pas subsister lorsque les entrées $\overline{S}$ et $\overline{R}$ reviennent à leur niveau (1)

Figure 5-9 Circuit intégré 74279 comprenant quatre bascules $\overline{S}\overline{R}$

Dans le C1 7475 à quatre bascules, voir ci-dessous, les variables primaires G appliquées aux entrées ENABLE jouent le même rôle que l'entrée primaire d'horloge T vue ci-dessus. L'entrée primaire D peut être toutefois ramenée à S et R, comme la figure 5-10 nous le montre.

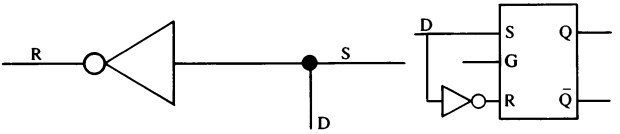
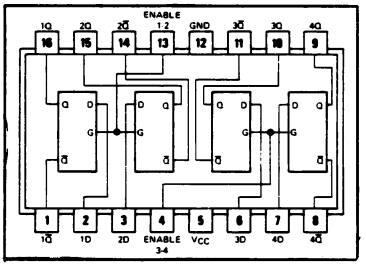


Figure 5-10

Cette bascule permet d'avoir à la sortie Q l'entrée D si le signal G appliquée à l'entrée ENABLE est à 1.

FUNCTION TABLE
(Each Latch)

INPUTS		OUTPUTS	
D	G	Q	$\overline{Q}$
L	H	L	H
H	H	H	L
X	L	Q_0	$\overline{Q}_0$



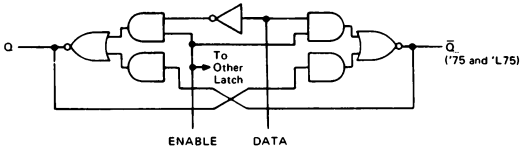


Figure 5-11 Table de fonctions, schéma de brochage et diagramme synoptique des fonctions du C1 7475.

5-4 LA BASCULE JK

Les bascules vues jusqu'à maintenant présentent des inconvénients. Tout d'abord, elles ont une combinaison interdite ou tout au moins un état qui n'est pas stable lorsque les deux entrées sont à 0. Lorsque le circuit revient au repos on ne sait pas dans quel état il se stabilisera. Un deuxième inconvénient est qu'on ne peut pas contrôler les temps de propagation dans le circuit. Pour les bascules avec horloge la sortie prendra donc l'état commandé à un moment qu'on ne peut pas contrôler durant l'*impulsion d'horloge*. On voudrait aussi une opération supplémentaire très utile, la complémentation, qui sera obtenue en faisant changer l'état de ce circuit sans être obligé, pour appliquer la commande adéquate, de connaître l'état précédent. On dispose pour ce faire d'une bascule appelée *bascule T*, cette bascule reinjecte les sorties à l'entrée. Le schéma de la figure 5-12 en donne le principe.

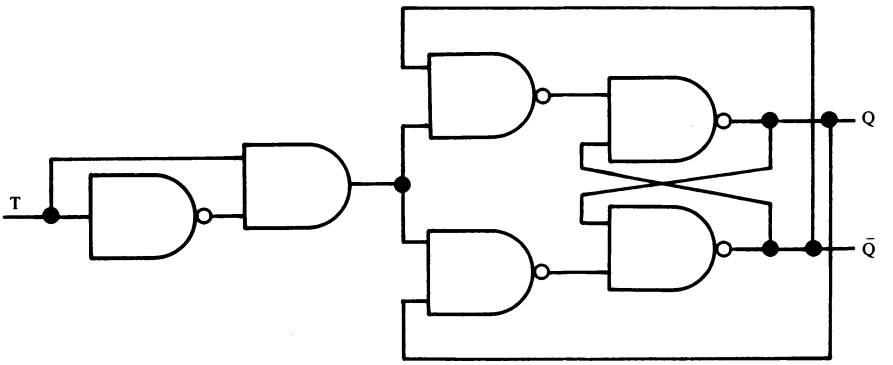
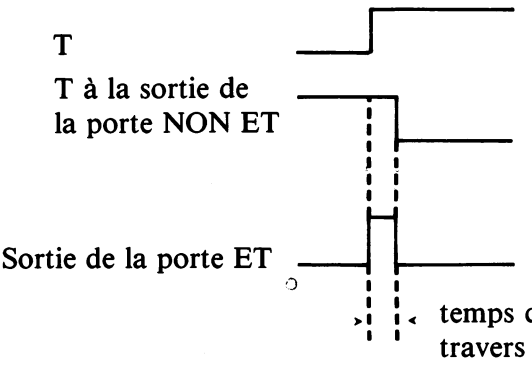


Figure 5-12 Bascule T sur flanc d'onde montant

Lorsque T passe à 1 ⎍ la sortie de la porte ET passe à 1 durant la propagation du signal T = 1 à travers la porte NON ET



Sans cette commande le circuit est instable.

$Q = 1$ et $T = 1$ commandent $\bar{Q}^{+} = 1$ et $Q^{+} = 0$

$Q = 0$ et $T = 1$ commandent $\bar{Q}^{+} = 0$ et $Q^{+} = 1$

Si $T = 0$ le circuit est stable et reste dans l'état où il se trouvait lorsque T était égal à 1.

On a bien un circuit qui bascule à chaque transition de T. On peut aussi considérer ce circuit comme un *diviseur de fréquence par deux*. Il revient au même état après deux impulsions successives de l'horloge T.

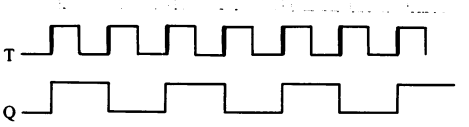


Figure 5-13 Diviseur de fréquence par deux

Ce circuit ne contrôle pas l'instant où la nouvelle valeur de la sortie apparaîtra. Pour éliminer cet inconvénient on a imaginé des circuits appelés *bascules JK*. Nous ne nous attarderons pas sur la synthèse de ce circuit, mais nous analyserons son principe de fonctionnement.

Principe de la notion maître-esclave

Reprenons à la figure 5-14 la bascule SRT de la figure 5-7.

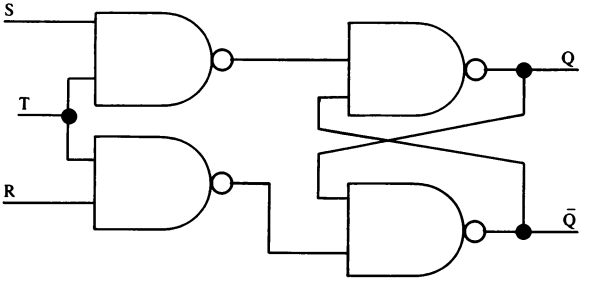


Figure 5-14 Bascule SRT

Plaçons deux circuits de ce type en cascade et inversons le signal d'horloge du deuxième. Voir la figure 5-15. Nous avons alors un circuit appelé *bascule maître-esclave*. Le premier circuit est le maître et le second l'esclave. Analysons le fonctionnement de ce circuit. Supposons que $S = 1$ et $R = 0$, pour $T = 0$ rien ne change à la sortie, la bascule maître reste à son état précédent et il en va de même pour la bascule esclave, pour $T = 1$ la deuxième bascule a une entrée d'horloge $T = 0$, elle est donc *bloquée* et ne change pas d'état.

S = T = 1, R = 0 entraînent Qm = 1 (la sortie 0 de la porte 1 commande Qm = 1) et $\bar{Q}m$ = 0.

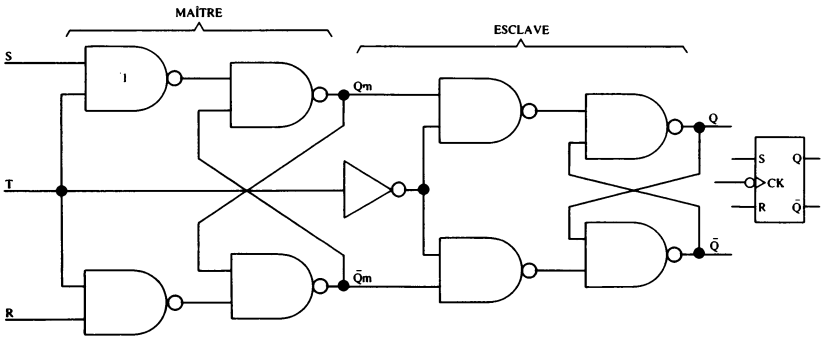


Figure 5-15 Bascule *maître-esclave*

Cet état reste stable tant que T = 1 et que les entrées R et S ne changent pas.

Lorsque T revient à 0, la bascule maître est bloquée et elle n'est plus sensible aux variations des entrées R et S. La bascule esclave devient active. Qm et $\bar{Q}m$ jouent le rôle de S et R de la bascule précédente et la bascule esclave transfère à la sortie Q le niveau logique de Qm. On a donc transfert à la sortie sur le *front d'onde descendant* de l'horloge. On peut résumer notre analyse par le schéma ci-dessous:

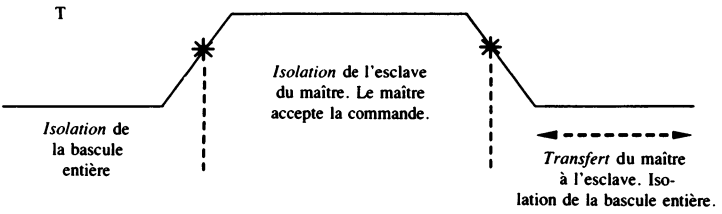


Figure 5-16

Pour obtenir la bascule JK il suffit de renvoyer les sorties aux entrées et on a *basculement* sur le front d'onde descendant de l'horloge lorsque J et K valent 1. Lorsque J et K valent 0 tous les deux, on a blocage du système et la bascule reste dans son état quel que soit le signal d'horloge. Pour J = 0 et K = 1 on commande Q = 0 après le signal d'horloge. Pour J = 1 et K = 0 on commande Q = 1 après le signal d'horloge.

La bascule JK obtenue est représentée à la figure 5-17.

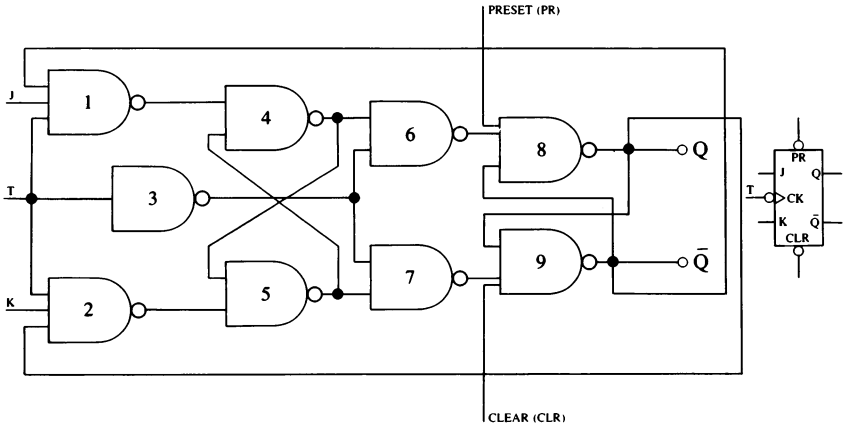


Figure 5-17 Bascule maître-esclave JK

Nous avons ajouté à ce schéma deux lignes Preset* et Clear**. Si Preset = 0, alors Q = 1 quels que soient le signal d'horloge et les entrées. Si Clear = 0, alors Q = 0 quels que soient le signal d'horloge et les entrées (dans le cas de la combinaison non stable, ceci ne persistera pas lorsque les entrées Preset et Clear reviendront à leur *niveau inactif*).

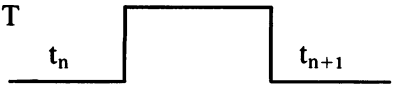
Les signaux Preset et Clear sont asynchrones, c'est-à-dire que leur commande entraîne immédiatement la sortie commandée prioritaire: ils passent par-dessus toutes les autres commandes. L'entrée Preset est parfois appelée Set.

La table de vérité suivante résume le fonctionnement d'une bascule JK.

t_n		t_{n+1}
J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\overline{Q_n}$

NdC * *Mise à 1 prereglee*
** *Remise à 0*

t_n et t_{n+1} désignent respectivement l'instant avant et après l'impulsion d'horloge,



Q_n est la valeur de Q à l'instant t_n.

Si J = K = 0 le système est bloqué. Q garde l'état précédent.

J = 0 et K = 1 commandent Q = 0

J = 1 et K = 0 commandent Q = 1

J = K = 1 le système bascule. Q prend donc la valeur $\overline{Q}_n$

De cette table de vérité, déduisons une autre table appelée *table d'excitation,* elle nous servira à synthétiser les circuits utilisant des bascules JK.

Cette table permet de déterminer les commandes d'entrée J et K à appliquer pour obtenir la sortie désirée connaissant la sortie avant l'impulsion d'horloge.

1er cas: $Q_n = 0$, on veut $Q_{n+1} = 0$

Selon la table de vérité, J = 0 et K = 1 commandent $Q_{n+1} = 0$ et J = K = 0 commandent $Q_{n+1} = Q_n = 0$ par hypothèse. D'où l'excitation à appliquer: J = 0, peu importe la valeur, 0 ou 1, de K. Dans un tel cas de valeur indifférente on porte la lettre X. D'où la première ligne de la table d'excitation ci-dessous.

2e cas: $Q_n = 0$, on veut $Q_{n+1} = 1$

Selon la table de vérité, J = 1 et K = 0 commandent $Q_{n+1} = 1$ et J = K = 1 commandent $Q_{n+1} = \overline{Q}_n = 1$ par hypothèse. D'où l'excitation à appliquer: J = 1, peu importe K. Ce qui donne la deuxième ligne de la table d'excitation.

3e cas: $Q_n = 1$, on veut $Q_{n+1} = 0$

Selon la table de vérité, J = 0 et K = 1 commandent $Q_{n+1} = 0$ et J = K = 1 commandent $Q_{n+1} = \overline{Q}_n = 0$ par hypothèse. D'où l'excitation à appliquer: K = 1, peu importe J. Ce qui donne la troisième ligne de la table d'excitation.

4e cas: $Q_n = 1$, on veut $Q_{n+1} = 1$

Selon la table de vérité, J = 1 et K = 0 commandent $Q_{n+1} = 1$ et J = K = 0 commandent $Q_{n+1} = Q_n = 1$ par hypothèse. D'où l'excitation à appliquer: K = 0, peu importe J. Ce qui donne la quatrième ligne de la table d'excitation.

Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Table d'excitation des bascules JK

L'analyse de ce circuit nous permet de faire aussi une autre remarque très importante: il faut que les valeurs des entrées J et K soient *stabilisées* avant l'impulsion d'horloge et le restent durant la *période active* de l'horloge, c'est-à-dire lorsque T = 1 dans notre cas. En effet, regardons sur la figure 5-17 ce qui se passe lorsque, pour T = 1, Q= 1 et $\overline{Q} = 0$, K, de niveau logique 0, passe à 1 un court instant et revient à 0. (Puisque $\overline{Q} = 0$ la valeur de J ne nous intéresse pas, car la sortie de la porte 1 est toujours 1). Pour K = 0 la sortie de la porte 2 est toujours 1, il n'y a donc pas de changement à la sortie de la bascule maître constituée des portes 4 et 5. La sortie de la porte 4 est 1, celle de la porte 5 est 0. Lorsque K passe à 1, la sortie de la porte 2 passe à 0, celle de 5 à 1 et celle de 4 à 0.

Lorsque K revient à 0, on revient dans le premier cas, car la bascule maître ne change pas d'état (porte 4 de sortie 0 et porte 5 de sortie 1). Lorsque le signal d'horloge disparaît, on aura Q = 0 et $\overline{Q} = 1$, donc un changement d'état. Le choix au départ de J = X et K = 0 était dicté par la volonté de garder la sortie à 1, donc d'avoir $Q_{n+1} = 1$ (voir la table d'excitation). Le transitoire K = 1 a provoqué un basculement et entraîné une sortie erronée. Il faut donc bien vérifier la stabilisation des entrées J et K avant d'envoyer une impulsion d'horloge. Ceci évitera de longues recherches de pannes ou d'erreurs dans les circuits. Pour contourner ces difficultés on a mis au point des bascules JK à *déclenchement sur front d'impulsion* (edge triggered). Dans ces circuits synchrones la période active de la commande a lieu sur le front d'impulsion (passage à un certain niveau de tension) du signal d'horloge et non durant tout ce signal).

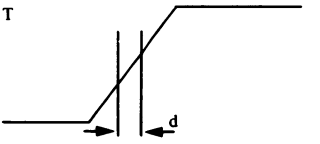


Figure 5-18 Durée d de commande de la bascule.

On dit parfois qu'on a un *couplage* CA au lieu d'un *couplage* CC, pour préciser que seule la variation du signal de commande d'horloge importe et non son niveau.

5-5 ÉTUDE DES BASCULES EN CIRCUITS INTÉGRÉS

Les planches suivantes donnent les configurations de brochage et les tables de vérité des principales bascules de la série 74xxx TTL.

Dans une référence, la lettre H indique un circuit *haute vitesse* et la lettre L désigne un circuit de *faible consommation de puissance*.

Suivant les circuits intégrés, chaque bascule a les deux entrées Clear et Preset, ou seulement l'une des deux. Les entrées Clear et Clock sont parfois communes à deux bascules. Les bascules JK peuvent avoir des portes ET, OU (74 H 71, 74 H 101, par exemple). Les portes ET de ces derniers modèles sont reliées entre elles et à l'entrée d'horloge. Voir les planches ci-dessous et le manuel de la société Texas Instruments pour d'autres précisions. Remarquons, en particulier, que la bascule 7470 comporte des entrées $\overline{J}$ et $\overline{K}$.

Cette dernière particularité permet de réaliser des bascules D (Data). Si l'on veut qu'une bascule JK serve d'élément de mise en mémoire d'une valeur binaire D (0 ou 1) à sa sortie Q, il faut, selon la table de vérité de la bascule JK, que J = D et K = $\overline{D}$.

La bascule D aura donc la configuration de la figure 5-19.

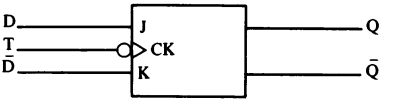


Figure 5-19 Bascule D

Si l'entrée $\overline{K}$ est disponible sur le circuit intégré, on obtient la figure 5-20.

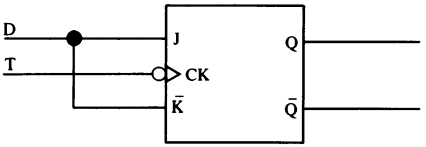


Figure 5-20 Câblage d'une bascule JK en type D

Le 7474 est par construction une bascule de type D.

Expliquons l'expression *DATA LOCKOUT* (*verrouillage de données*). On a vu plus haut qu'il fallait stabiliser les entrées J et K avant d'envoyer une impulsion d'horloge sinon on pouvait avoir une sortie erronée. On a contourné cette difficulté en insérant un léger retard dans la bascule (voir les CI 74110 et 74111) qui permet de ne tenir compte des entrées J et K que durant un très court instant au début du signal d'horloge (voir la figure 5-21). J et K peuvent ensuite changer, sans influencer la sortie.

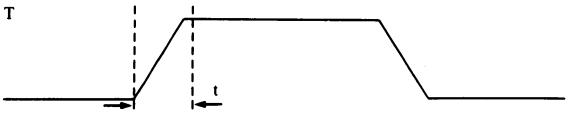


Figure 5-21 Temps t de verrouillage des données.

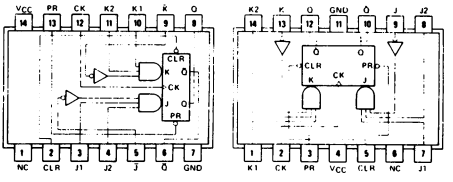
La grande variété de bascules intégrées JK en circulation répond aux nombreuses applications de ce type de circuits. Parmi celles-ci nous étudierons, entre autres, les *compteurs*, les *registres à décalage* et les *mémoires temporaires*. Nous envisageons tout d'abord quelques petites applications pratiques.

AND-GATED J-K POSITIVE-EDGE-TRIGGERED FLIP-FLOPS WITH PRESET AND CLEAR

7470 FUNCTION TABLE

INPUTS					OUTPUTS	
PRESET	CLEAR	CLOCK	J	K	Q	$\overline{Q}$
L	H	L	X	X	H	L
H	L	L	X	X	L	H
L	L	X	X	X	L*	L*
H	H	↑	L	L	Q_0	$\overline{Q}_0$
H	H	↓	H	L	H	L
H	H	↑	L	H	L	H
H	H	↑	H	H	TOGGLE	
H	H	L	X	X	Q_0	$\overline{Q}_0$

positive logic: $J = J1 \cdot J2 \cdot \overline{J}$
$K = K1 \cdot K2 \cdot \overline{K}$
If inputs $\overline{J}$ and $\overline{K}$ are not used, they must be grounded.
Preset or clear function can occur only when the clock input is low.



SN5470 (J) SN7470 (J, N) SN5470 (W)

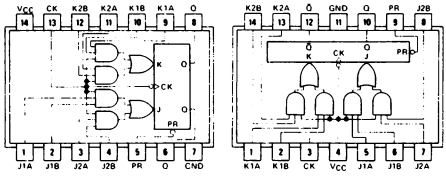
AND-OR-GATED J-K MASTER-SLAVE FLIP-FLOPS WITH PRESET

74H71

FUNCTION TABLE

INPUTS				OUTPUTS	
PRESET	CLOCK	J	K	Q	$\bar{Q}$
L	X	X	X	H	L
H	⎍	L	L	Q_0	$\bar{Q}_0$
H	⎍	H	L	H	L
H	⎍	L	H	L	H
H	⎍	H	H	TOGGLE	

positive logic: J = (J1A·J1B)+(J2A·J2B)
K = (K1A·K1B)+(K2A·K2B)



SN54H71 (J) SN74H71 (J N) SN54H71 (W)

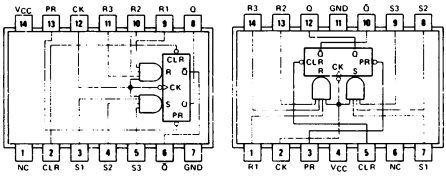
AND-GATED R-S MASTER-SLAVE FLIP-FLOPS WITH PRESET AND CLEAR

74L71

FUNCTION TABLE

INPUTS					OUTPUTS	
PRESET	CLEAR	CLOCK	S	R	Q	$\bar{Q}$
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H*	H*
H	H	⎍	L	L	Q_0	$\bar{Q}_0$
H	H	⎍	H	L	H	L
H	H	⎍	L	H	L	H
H	H	⎍	H	H	INDETERMINATE	

positive logic: R = R1·R2·R3
S = S1·S2·S3



SN54L71 (J) SN74L71 (J, N) SN54L71 (T)

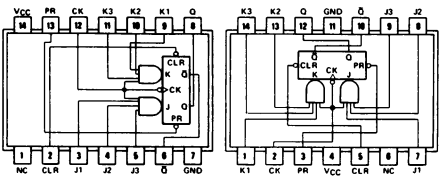
AND-GATED J-K MASTER-SLAVE FLIP-FLOPS WITH PRESET AND CLEAR

7472

FUNCTION TABLE

INPUTS					OUTPUTS	
PRESET	CLEAR	CLOCK	J	K	Q	$\bar{Q}$
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H*	H*
H	H	⎍	L	L	Q_0	$\bar{Q}_0$
H	H	⎍	H	L	H	L
H	H	⎍	L	H	L	H
H	H	⎍	H	H	TOGGLE	

positive logic: J = J1·J2·J3; K1·K2·K3



SN5472 (J) SN7472 (J, N) SN5472 (W)
SN54H72 (J) SN74H72 (J, N) SN54H72 (W)
SN54L72 (J) SN74L72 (J, N) SN54L72 (T)

DUAL J-K FLIP-FLOPS WITH CLEAR

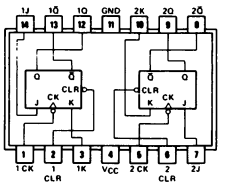
7473

'73, 'H73, 'L73
FUNCTION TABLE

INPUTS				OUTPUTS	
CLEAR	CLOCK	J	K	Q	$\bar{Q}$
L	X	X	X	L	H
H	⎍	L	L	Q_0	$\bar{Q}_0$
H	⎍	H	L	H	L
H	⎍	L	H	L	H
H	⎍	H	H	TOGGLE	

'LS73A
FUNCTION TABLE

INPUTS				OUTPUTS	
CLEAR	CLOCK	J	K	Q	$\bar{Q}$
L	X	X	X	L	H
H	↓	L	L	Q_0	$\bar{Q}_0$
H	↓	H	L	H	L
H	↓	L	H	L	H
H	↓	H	H	TOGGLE	
H	H	X	X	Q_0	$\bar{Q}_0$



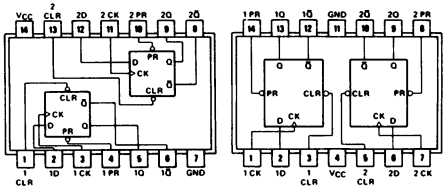
SN5473 (J, W) SN7473 (J, N)
SN54H73 (J, W) SN74H73 (J, N)
SN54L73 (J, T) SN74L73 (J, N)
SN54LS73A (J, W) SN74LS73A (J, N)

DUAL D-TYPE POSITIVE-EDGE-TRIGGERED FLIP-FLOPS WITH PRESET AND CLEAR

7474

FUNCTION TABLE

INPUTS				OUTPUTS	
PRESET	CLEAR	CLOCK	D	Q	$\bar{Q}$
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	H*	H*
H	H	↑	H	H	L
H	H	↑	L	L	H
H	H	L	X	Q_0	$\bar{Q}_0$



SN5474 (J) SN7474 (J, N) SN5474 (W)
SN54H74 (J) SN74H74 (J, N) SN54H74 (W)
SN54L74 (J) SN74L74 (J, N) SN54L74 (T)
SN54LS74A (J, W) SN74LS74A (J, N)
SN54S74 (J, W) SN74S74 (J, N)

DUAL J-K FLIP-FLOPS WITH PRESET AND CLEAR

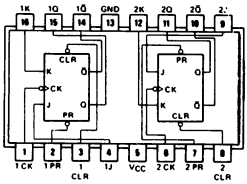
7476

'76, 'H76
FUNCTION TABLE

INPUTS					OUTPUTS	
PRESET	CLEAR	CLOCK	J	K	Q	$\bar{Q}$
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H*	H*
H	H	⊓	L	L	Q_0	$\bar{Q}_0$
H	H	⊓	H	L	H	L
H	H	⊓	L	H	L	H
H	H	⊓	H	H	TOGGLE	

'LS76A
FUNCTION TABLE

INPUTS					OUTPUTS	
PRESET	CLEAR	CLOCK	J	K	Q	$\bar{Q}$
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H*	H*
H	H	↓	L	L	Q_0	$\bar{Q}_0$
H	H	↓	H	L	H	L
H	H	↓	L	H	L	H
H	H	↓	H	H	TOGGLE	
H	H	H	X	X	Q_0	Q_0



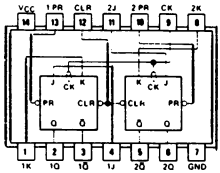
SN5476 (J, W) SN7476 (J, N)
SN54H76 (J, W) SN74H76 (J, N)
SN54LS76A (J, W) SN74LS76A (J, N)

DUAL J-K FLIP-FLOPS WITH PRESET, COMMON CLEAR, AND COMMON CLOCK

7478

'H78, 'L78
FUNCTION TABLE

INPUTS					OUTPUTS	
PRESET	CLEAR	CLOCK	J	K	Q	$\bar{Q}$
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H*	H*
H	H	⊓	L	L	Q_0	$\bar{Q}_0$
H	H	⊓	H	L	H	L
H	H	⊓	L	H	L	H
H	H	⊓	H	H	TOGGLE	

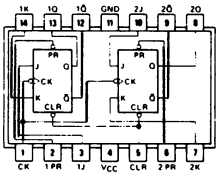


SN54H78(J,W) SN74H78(J,N)

See pages 6-50 and 6-54

'LS78A
FUNCTION TABLE

INPUTS					OUTPUTS	
PRESET	CLEAR	CLOCK	J	K	Q	$\bar{Q}$
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H*	H*
H	H	⊓	L	L	Q_0	$\bar{Q}_0$
H	H	⊓	H	L	H	L
H	H	⊓	L	H	L	H
H	H	⊓	H	H	TOGGLE	
H	H	H	X	X	Q_0	$\bar{Q}_0$



SN54L78(J,T) SN74L78(J,N)
SN54LS78A(J,W) SN74LS78A(J,N)

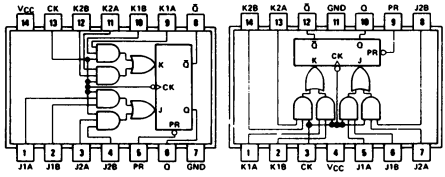
AND-OR-GATED J-K NEGATIVE-EDGE-TRIGGERED FLIP-FLOPS WITH PRESET

74101

FUNCTION TABLE

INPUTS				OUTPUTS	
PRESET	CLOCK	J	K	Q	$\bar{Q}$
L	X	X	X	H	L
H	↓	L	L	Q_0	$\bar{Q}_0$
H	↓	H	L	H	L
H	↓	L	H	L	H
H	↓	H	H	TOGGLE	
H	H	X	X	Q_0	$\bar{Q}_0$

positive logic: $J = (J1A \cdot J1B) + (J2A \cdot J2B)$
$K = (K1A \cdot K1B) + (K2A \cdot K2B)$



SN54H101 (J) SN74H101 (J, N) SN54H101 (W)

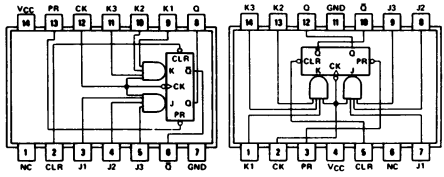
AND-GATED J-K NEGATIVE-EDGE-TRIGGERED FLIP-FLOPS WITH PRESET AND CLEAR

74102

FUNCTION TABLE

INPUTS					OUTPUTS	
PRESET	CLEAR	CLOCK	J	K	Q	$\bar{Q}$
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H*	H*
H	H	↓	L	L	Q_0	$\bar{Q}_0$
H	H	↓	H	L	H	L
H	H	↓	L	H	L	H
H	H	↓	H	H	TOGGLE	
H	H	H	X	X	Q_0	$\bar{Q}_0$

positive logic: $J = J1 \cdot J2 \cdot J3$
$K = K1 \cdot K2 \cdot K3$



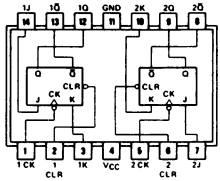
SN54H102 (J) SN74H102 (J,N) SN54H102 (W)

DUAL J-K NEGATIVE-EDGE-TRIGGERED FLIP-FLOPS WITH CLEAR

74103

FUNCTION TABLE

INPUTS				OUTPUTS	
CLEAR	CLOCK	J	K	Q	$\bar{Q}$
L	X	X	X	L	H
H	↓	L	L	Q_0	$\bar{Q}_0$
H	↓	H	L	H	L
H	↓	L	H	L	H
H	↓	H	H	TOGGLE	
H	H	X	X	Q_0	$\bar{Q}_0$



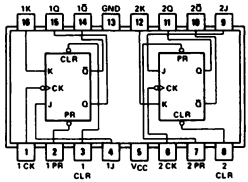
SN54H103 (J, W) SN74H103 (J, N)

DUAL J-K NEGATIVE-EDGE-TRIGGERED FLIP-FLOPS WITH PRESET AND CLEAR

74106

FUNCTION TABLE

INPUTS					OUTPUTS	
PRESET	CLEAR	CLOCK	J	K	Q	$\bar{Q}$
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H*	H*
H	H	↓	L	L	Q_0	$\bar{Q}_0$
H	H	↓	H	L	H	L
H	H	↓	L	H	L	H
H	H	↓	H	H	TOGGLE	
H	H	H	X	X	Q_0	$\bar{Q}_0$



SN54H106 (J, W) SN74H106 (J, N)

DUAL J-K FLIP-FLOPS WITH CLEAR

74107

'107

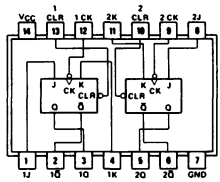
FUNCTION TABLE

INPUTS				OUTPUTS	
CLEAR	CLOCK	J	K	Q	$\bar{Q}$
L	X	X	X	L	H
H	⎍	L	L	Q_0	$\bar{Q}_0$
H	⎍	H	L	H	L
H	⎍	L	H	L	H
H	⎍	H	H	TOGGLE	

'LS107A

FUNCTION TABLE

INPUTS				OUTPUTS	
CLEAR	CLOCK	J	K	Q	$\bar{Q}$
L	X	X	X	L	H
H	↓	L	L	Q_0	$\bar{Q}_0$
H	↓	H	L	H	L
H	↓	L	H	L	H
H	↓	H	H	TOGGLE	
H	H	X	X	Q_0	$\bar{Q}_0$



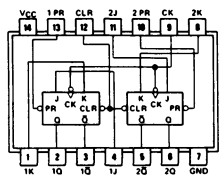
SN54107 (J) SN74107 (J, N)
SN54LS107A (J) SN74LS107A (J, N)

DUAL J-K NEGATIVE-EDGE-TRIGGERED FLIP-FLOPS WITH PRESET, COMMON CLEAR, AND COMMON CLOCK

74108

FUNCTION TABLE

INPUTS					OUTPUTS	
PRESET	CLEAR	CLOCK	J	K	Q	$\bar{Q}$
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H*	H*
H	H	↓	L	L	Q_0	$\bar{Q}_0$
H	H	↓	H	L	H	L
H	H	↓	L	H	L	H
H	H	↓	H	H	TOGGLE	
H	H	H	X	X	Q_0	$\bar{Q}_0$



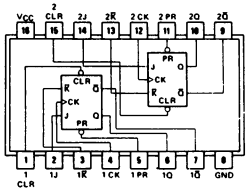
SN54H108 (J, W) SN74H108 (J, N)

DUAL J-$\overline{K}$ POSITIVE-EDGE-TRIGGERED FLIP-FLOPS WITH PRESET AND CLEAR

74109

FUNCTION TABLE

INPUTS					OUTPUTS	
PRESET	CLEAR	CLOCK	J	$\overline{K}$	Q	$\overline{Q}$
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H*	H*
H	H	↓	L	L	L	H
H	H	↓	H	L	TOGGLE	
H	H	↓	L	H	Q_0	$\overline{Q}_0$
H	H	↓	H	H	H	L
H	H	L	X	X	Q_0	$\overline{Q}_0$



SN54109 (J, W) SN74109 (J, N)
SN54LS109A (J, W) SN74LS109A (J, N)

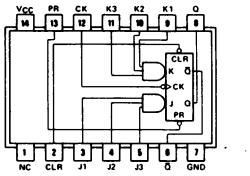
AND-GATED J-K MASTER-SLAVE FLIP-FLOPS WITH DATA LOCKOUT

74110

FUNCTION TABLE

INPUTS					OUTPUTS	
PRESET	CLEAR	CLOCK	J	K	Q	$\overline{Q}$
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H*	H*
H	H	⊓	L	L	Q_0	$\overline{Q}_0$
H	H	⊓	H	L	H	L
H	H	⊓	L	H	L	H
H	H	⊓	H	H	TOGGLE	

positive logic: J = J1·J2·J3
K = K1·K2·K3



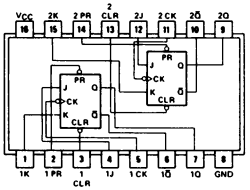
SN54110 (J, W) SN74110 (J, N)

DUAL J-K MASTER-SLAVE FLIP-FLOPS WITH DATA LOCKOUT

74111

FUNCTION TABLE

INPUTS					OUTPUTS	
PRESET	CLEAR	CLOCK	J	K	Q	$\overline{Q}$
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H*	H*
H	H	⊓	L	L	Q_0	$\overline{Q}_0$
H	H	⊓	H	L	H	L
H	H	⊓	L	H	L	H
H	H	⊓	H	H	TOGGLE	



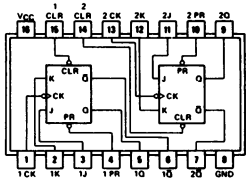
SN54111 (J, W) SN74111 (J, N)

DUAL J-K NEGATIVE-EDGE-TRIGGERED FLIP-FLOPS WITH PRESET AND CLEAR

74112

FUNCTION TABLE

INPUTS					OUTPUTS	
PRESET	CLEAR	CLOCK	J	K	Q	$\bar{Q}$
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H*	H*
H	H	↓	L	L	Q_0	$\bar{Q}_0$
H	H	↓	H	L	H	L
H	H	↓	L	H	L	H
H	H	↓	H	H	TOGGLE	
H	H	H	X	X	Q_0	$\bar{Q}_0$



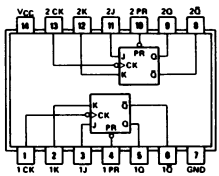
SN54LS112A (J, W) SN74LS112A (J,N)
SN54S112 (J, W) SN74S112 (J, N)

DUAL J-K NEGATIVE-EDGE-TRIGGERED FLIP-FLOPS WITH PRESET

74113

FUNCTION TABLE

INPUTS				OUTPUTS	
PRESET	CLOCK	J	K	Q	$\bar{Q}$
L	X	X	X	H	L
H	↓	L	L	Q_0	$\bar{Q}_0$
H	↓	H	L	H	L
H	↓	L	H	L	H
H	↓	H	H	TOGGLE	
H	H	X	X	Q_0	$\bar{Q}_0$



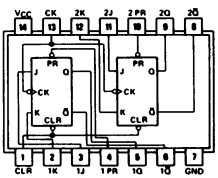
SN54LS113A (J, W) SN74LS113A (J, N)
SN54S113 (J, W) SN74S113 (J, N)

DUAL J-K NEGATIVE-EDGE-TRIGGERED FLIP-FLOPS WITH PRESET, COMMON CLEAR, AND COMMON CLOCK

74114

FUNCTION TABLE

INPUTS					OUTPUTS	
PRESET	CLEAR	CLOCK	J	K	Q	$\bar{Q}$
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H*	H*
H	H	↓	L	L	Q_0	$\bar{Q}_0$
H	H	↓	H	L	H	L
H	H	↓	L	H	L	H
H	H	↓	H	H	TOGGLE	
H	H	H	X	X	Q_0	$\bar{Q}_0$



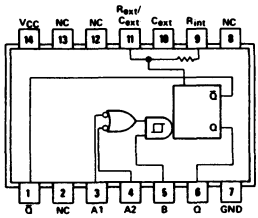
SN54LS114A (J, W) SN74LS114A (J, N)
SN54S114 (J, W) SN74S114 (J,N)

MONOSTABLE MULTIVIBRATORS

74121

FUNCTION TABLE

INPUTS			OUTPUTS	
A1	A2	B	Q	$\bar{Q}$
L	X	H	L	H
X	L	H	L	H
X	X	L	L	H
H	H	X	L	H
H	↓	H	⎍	⎎
↓	H	H	⎍	⎎
↓	↓	H	⎍	⎎
L	X	↑	⎍	⎎
X	L	↑	⎍	⎎



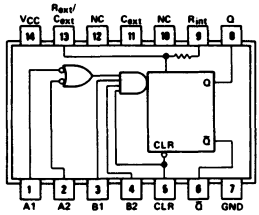
SN54121 (J, W) SN74121 (J, N)
SN54L121 (J, T) SN74L121 (J, N)

RETRIGGERABLE MONOSTABLE MULTIVIBRATORS WITH CLEAR

74122

FUNCTION TABLE

INPUTS					OUTPUTS	
CLEAR	A1	A2	B1	B2	Q	$\bar{Q}$
L	X	X	X	X	L	H
X	H	H	X	X	L	H
X	X	X	L	X	L	H
X	X	X	X	L	L	H
H	L	X	↑	H	⎍	⎎
H	L	X	H	↑	⎍	⎎
H	X	L	↑	H	⎍	⎎
H	X	L	H	↑	⎍	⎎
H	H	↓	H	H	⎍	⎎
H	↓	↓	H	H	⎍	⎎
H	↓	H	H	H	⎍	⎎
↑	L	X	H	H	⎍	⎎
↑	X	L	H	H	⎍	⎎



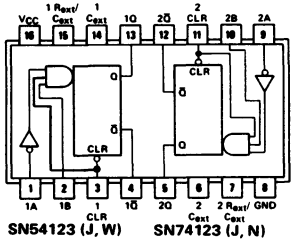
SN54122 (J, W) SN74122 (J, N)
SN54L122 (J, T) SN74L122 (J, N)
SN54LS122 (J, W) SN74LS122 (J, N)

DUAL RETRIGGERABLE MONOSTABLE MULTIVIBRATORS WITH CLEAR

74123

FUNCTION TABLE

INPUTS			OUTPUTS	
CLEAR	A	B	Q	$\bar{Q}$
L	X	X	L	H
X	H	X	L	H
X	X	L	L	H
H	L	↑	⎍	⎎
H	↓	H	⎍	⎎
↑	L	H	⎍	⎎



SN54123 (J, W) SN74123 (J, N)
SN54L123 (J) SN74L123 (J, N)
SN54LS123 (J, W) SN74LS123 (J, N)

5-6 APPLICATIONS DES BASCULES

5-6-1 ÉLIMINATEUR DES EFFETS DES REBONDISSEMENTS D'UN RELAIS

Voir la figure 5-22. La *lame* du relais est soit en A, soit en B. L'un de ces deux pôles met une entrée de la porte NON-ET correspondante à la masse, donc à un 0 logique et sa sortie à 1. Il se traduit, après la commutation, des *rebondissements* de la lame d'un pôle vers l'autre.

Il ne faut pas, au cours de ces rebondissements, que la lame vienne toucher l'autre pôle. Le circuit de la figure 5-22 élimine les conséquences fâcheuses des rebondissements. Au premier contact, la bascule change d'état. Rappelons-nous qu'en technologie TTL une entrée ouverte équivaut à un niveau logique 1 et une mise à la masse au niveau logique 0.

Les rebondissements n'ont aucun effet sur la sortie. Si, par exemple, on a contact en B ($B = 0$, $A = 1$), alors $\bar{Q} = 1$ et $Q = 0$. Au cours du rebondissement ($A = B = 1$) $\bar{Q}$ vaut toujours 1 ($Q = 0$). Pour les sorties, rien n'a donc changé et rien ne changera encore lorsque la lame reviendra en B.

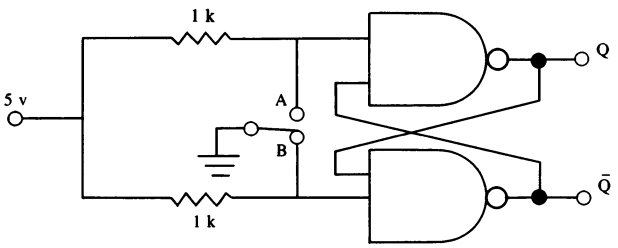


Figure 5-22 Éliminateur des effets des rebondissements

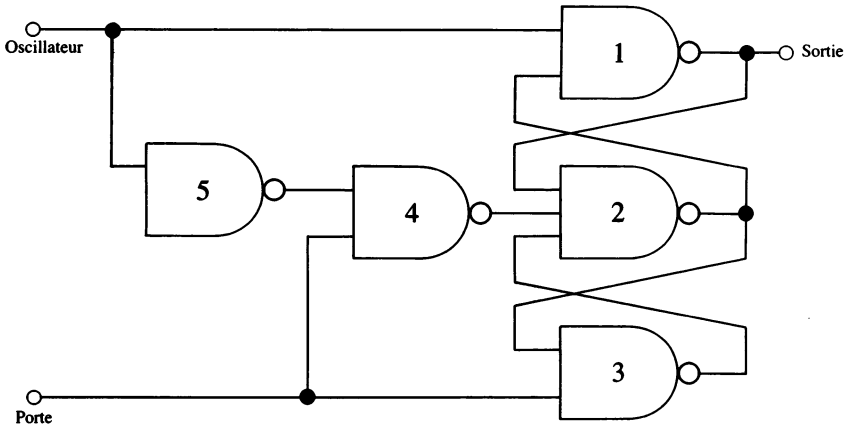
5-6-2 OSCILLATEUR À PORTE DE COMMANDE

Voir la figure 5-23. L'*oscillateur* fournit une onde carrée. Si l'entrée «PORTE» est à 0, la sortie de 3 et celle de 4 sont à 1. Lorsque l'oscillateur est à 0, la sortie de 1 est à 1 et celle de 2 à 0 (trois 1 à l'entrée), la sortie de 1 est donc toujours à 1 quelle que soit la valeur du signal de l'oscillateur. la sortie ne varie pas.

Lorsque le signal sur la porte est à 1 et le signal de l'oscillateur à 0, la sortie de 4 passe à 0, celle de 2 à 1 et celle de 3 à 0. La bascule constituée des portes 2 et 3 garde en mémoire, à la sortie de 2, la valeur 1 de l'entrée PORTE. Le signal de l'oscillateur apparaîtra inversé à la sortie de 1.

Lorsque la porte passe à 0 le signal d'horloge passe, s'il existe, et on revient à l'état initial lorsque le signal del'oscillateur est à 0. Si on a un

oscillateur *calibré* à 1MHz (T = 1μs), par exemple, on peut compter les impulsions passées durant l'*ouverture de la porte* et en déduire la durée de son ouverture. (Mesure d'un intervalle de temps ou d'une *période*.)



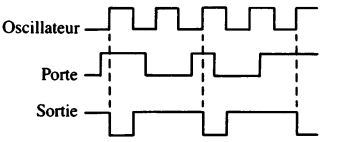


Figure 5-23 Oscillateur à porte de commande et son *chronogramme*

5-6-3 GÉNÉRATEUR D'UNE SIMPLE IMPULSION

On a souvent besoin de générer sur commande une simple impulsion calibrée par une horloge, pour un transfert en mémoire par exemple. C'est ce que réalise l'implantation de la figure 5-24.

L'impulsion G remet à 0 les deux bascules A et B, donc $Q_A = \overline{0}$ et $\overline{Q}_B = 1$. Cette impulsion asynchrone est prioritaire sur les autres entrées. Lorsque G = 0 le signal d'horloge n'a aucun effet sur le circuit. À l'état initial G = 1, $Q_A = 0$, $\overline{Q}_B = 1$. A bascule sur le front descendant de l'impulsion d'horloge donc $Q_A = 1$ ($J_A = K_A = 1$, bascule de type T) la sortie devient 0 ($Q_A = 1$, $\overline{Q}_B = 1$). B bascule sur le front montant de l'impulsion d'horloge qui devient le front descendant de B ($J_B = K_B = 1$) $\overline{Q}_B$ devient 0 et la sortie devient 1. À l'impulsion d'horloge suivante on a $J_A = 1$, $\overline{Q}_A = K_A = 0$, A reste donc dans le même état ($Q_A = 1$) ainsi que B ($J_B = 1$, $\overline{Q}_B = K_B = 0$): on ne laisse donc passer qu'une impulsion d'horloge. Pour faire repartir le système il faut le remettre à 0 à l'aide d'une impulsion sur G.

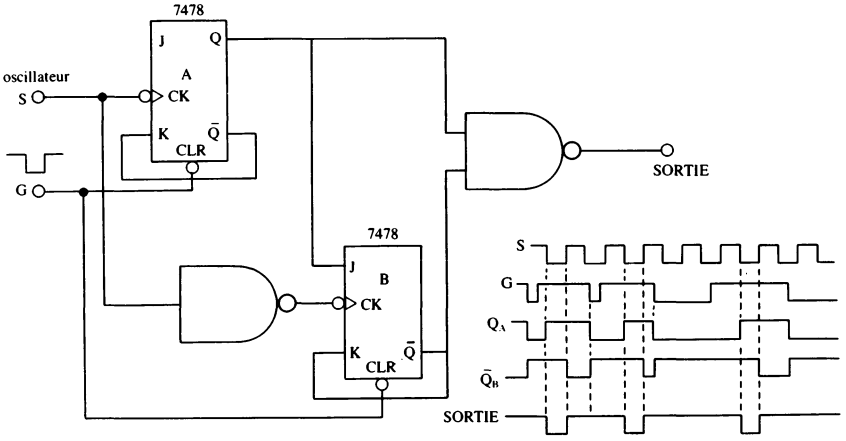


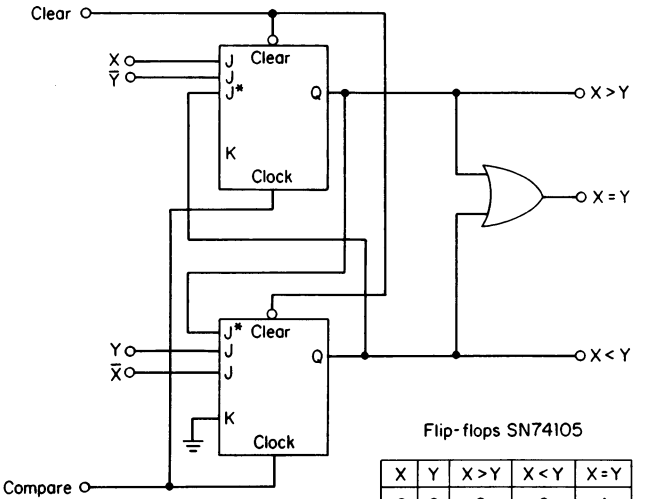
Figure 5-24 Générateur de simple impulsion et chronogramme

5-6-4 COMPARATEUR SÉRIE DE NOMBRES

On peut implanter une large variété de fonctions de décodage série, de comparaison et d'horloge à l'aide de bascules. La figure 5-25 représente un comparateur série.

Le fonctionnement de ce circuit est le suivant. Une impulsion initiale appliquée aux broches Clear remet à zéro les deux sorties Q. Toutes les entrées possibles avant le prélèvement des trois sorties X > Y, X = Y et X < Y sont définies par la table de vérité une fois l'impulsion de comparaison appliquée. (Remarque: Le SN 74102 est une bascule sur front descendant

d'impulsion.) Si deux mots, A et B, sont appliqués en série au comparateur à deux bits, la sortie X = Y reste au niveau logique 1 jusqu'à ce que deux bits des mots A et B soient différents. À cet instant, selon le cas, soit X > Y, soit X < Y sera au niveau logique 1. La connexion des deux sorties $\overline{Q}$ aux entrées J* verrouille le système lorsque X > Y ou X < Y est à 1, les sorties restent dans cet état jusqu'à l'application d'une impulsion de remise à zéro — (Application IF X > Y ou IF X < Y du calculateur HP).

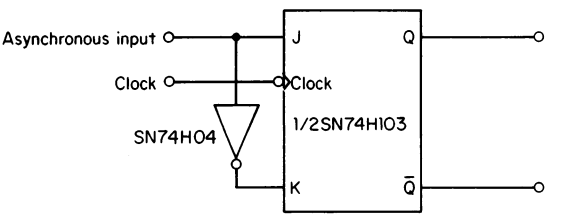


X	Y	X>Y	X<Y	X=Y
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1

Figure 5-25 Comparateur série de nombres (bascules JK 74105)

5-6-5 SYNCHRONISATEUR D'ENTRÉE ASYNCHRONE

Certains dispositifs nécessitent la synchronisation des données à une horloge. La figure 5-26 montre comment implanter une telle synchronisation à l'aide d'une bascule maître-esclave ou d'une bascule sur front d'impulsion. Voici la différence: la bascule synchronisera l'entrée des données de largeur plus grande que 20 ns présentes lorsque l'horloge sera à 1 alors que la bascule sur front d'impulsion synchronisera les données présentes à l'instant du front d'impulsion d'horloge l'activant.



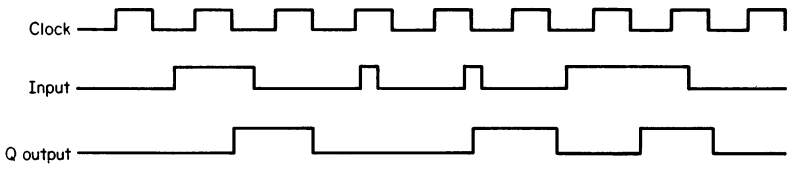


Figure 5-26 Synchronisateur d'entrée asynchrone et chronogramme.

5-6-6 GÉNÉRATEUR D'UNE SIMPLE IMPULSION SYNCHRONISÉE UNIFORME

La figure 5-27 donne une variante du générateur d'une simple impulsion. Les impulsions de sortie sont synchronisées avec l'horloge et toujours de la même façon. En général, l'entrée doit être au moins aussi large que l'impulsion d'horloge sinon elle n'est pas reconnue sauf si elle apparaît sur le front descendant d'impulsion d'horloge. On peut implanter cette fonction à l'aide de bascules JK, mais dans ce cas le circuit reconnaîtra des entrées de largeur supérieures à 20 ns pendant que le signal d'horloge est au niveau 1.

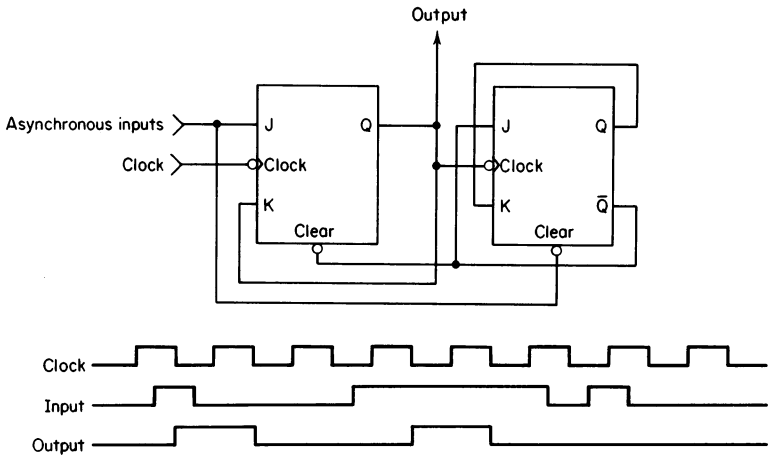


Figure 5-27 Générateur d'une simple impulsion synchronisée uniforme et chronogramme (bascules 74103)

5-6-7 GÉNÉRATEUR DE RAFALE D'IMPULSIONS SYNCHRONISÉES

La figure 5-28 illustre un générateur de rafale d'impulsions plus simple que celui de la figure 5-23. La sortie Q est un train d'impulsions soutenu tant que l'entrée asynchrone est au niveau logique 1. Les impulsions de sortie ont la même largeur que la durée du niveau logique 0 de l'horloge. Comme

tous les autres circuits utilisant des bascules maître-esclave, le circuit reconnaît des entrées de largeur supérieure à 20 ns lorsque l'horloge vaut 1. Cette particularité de la bascule maître-escalve permet d'utiliser ce circuit comme *détecteur de* 1 ou *d'erreurs*. Le chronogramme de la figure 5-28 illustre le fonctionnement de ce circuit avec une bascule maître-esclave et une horloge fournissant une fenêtre d'entrée.

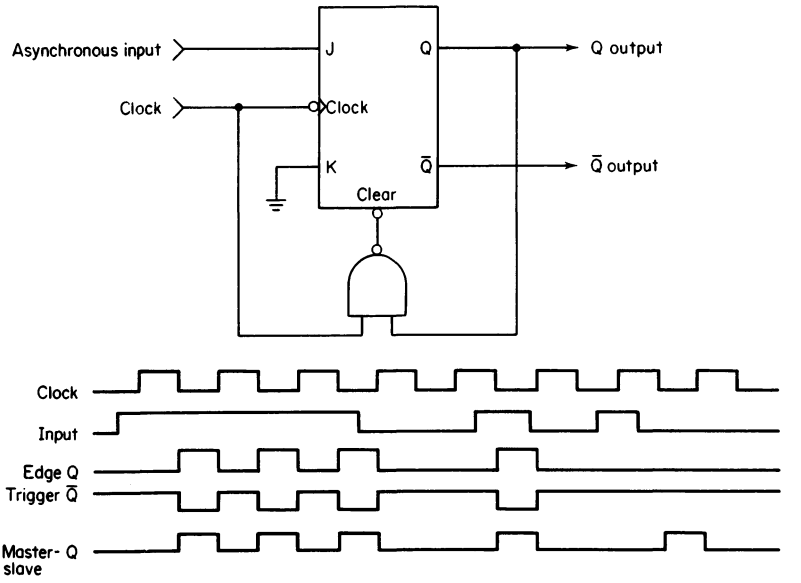


Figure 5-28 Générateur de rafale d'impulsions synchronisées (bascules 74103) et chronogramme

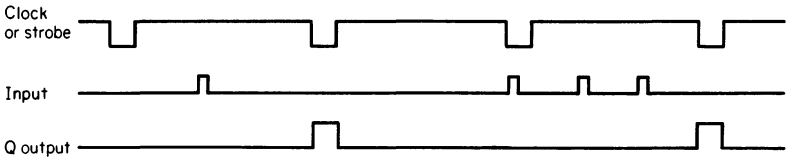


Figure 5-29 Chronogramme du circuit de la figure 5-28 comme détecteur de 1

5-7 COMPTEURS

5-7-1 COMPTEURS ASYNCHRONES MODULO 2^n

Nous avons déjà vu que la fréquence de sortie d'une bascule de type T vaut la moitié de celle de la commande. Soit une bascule JK avec, par exemple,

J = K = 1, c'est-à-dire les entrées J et K ouvertes. Les formes d'onde sont données à la figure 5-30. La valeur Q de la bascule apparaît à la sortie sur le front descendant d'impulsion d'horloge. Pour J = K = 1 on a $Q_{n+1} = \bar{Q}_n$.

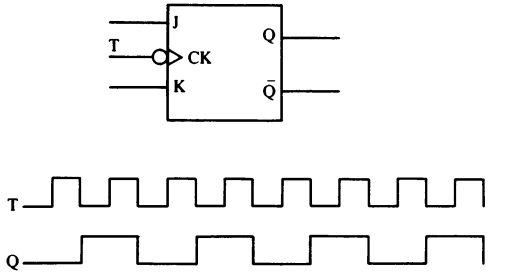


Figure 5-30 Formes d'onde de sortie d'une bascule JK

La figure 5-31 représente le montage et les sorties de deux telles bascules en cascade, la deuxième étant commandée par la sortie Q de la première. On a donc à la sortie Q_B une division par quatre de la fréquence T_A.

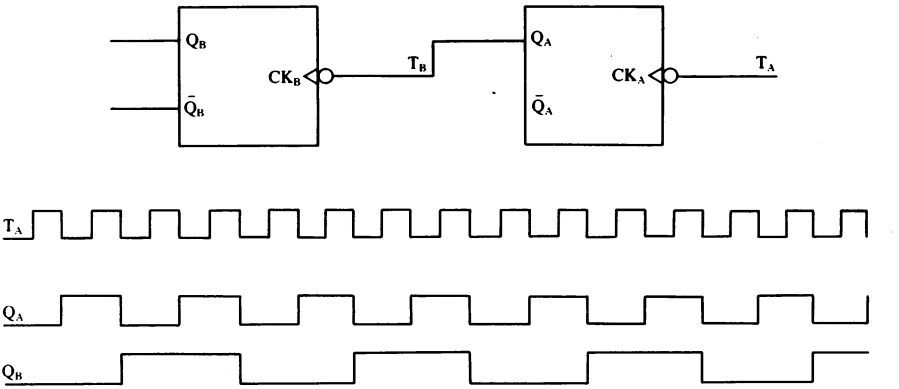


Figure 5-31 Diviseur par quatre de la fréquence

Si on place trois bascules JK suivant le schéma de la figure 5-31 on aura une division par huit, si on en place quatre on aura une division par seize et si on en place n on aura une division par 2^n.

Prenons, par exemple, trois bascules d'état initial Q = 0 commandées par une impulsion sur l'entrée CLEAR (remise à zéro, CLR). Le schéma et les formes d'onde apparaissent à la figure 5-32.

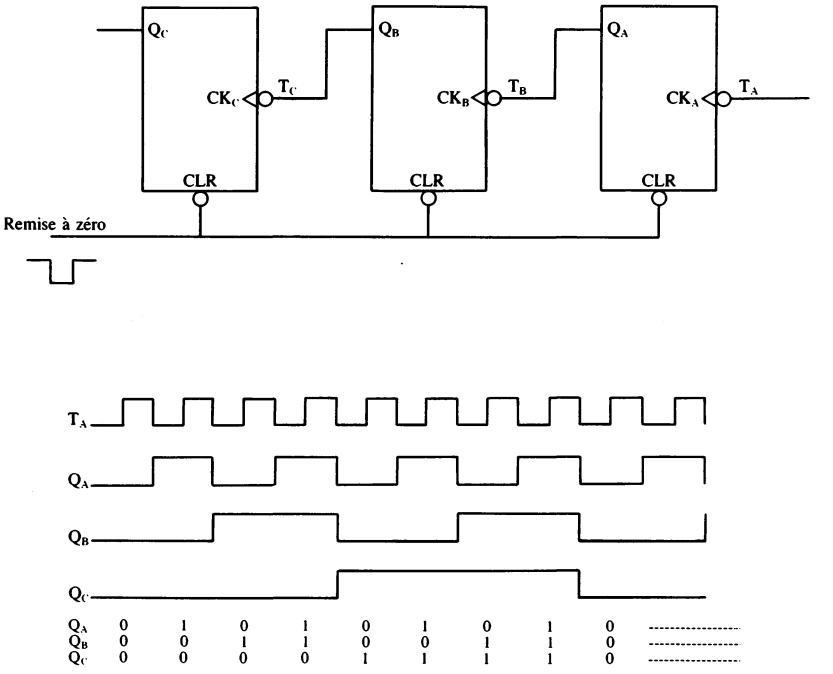


Figure 5-32 Compteur octal

Attribuons respectivement les symboles logiques 1 et 0 aux tensions haute et basse des sorties Q_A, Q_B et Q_C. Les états successifs de ce compteur relevés à ces sorties sont dès lors:

Q_C	Q_B	Q_A	
0	0	0	état initial
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

On a un compteur octal, c'est-à-dire un compteur à huit états représentés par les nombres 0 à 7 en binaire. Ce compteur part de l'état initial 000 (0), compte en binaire naturel jusqu'à 111 (7) en croissant, revient à 000 et recommence. On a un compteur *progressif*.

Si on avait pris les sorties Q comme référence, on aurait eu:

$\bar{Q}_C$	$\bar{Q}_B$	$\bar{Q}_A$
1	1	1
1	1	0
1	0	1
1	0	0
0	1	1
0	1	0
0	0	1
0	0	0

Un tel compteur est dit *régressif.*

Pour afficher l'état de tels compteurs, prenons l'affichage à sept segments à décodeur incorporé. Voir le montage de principe ci-dessous (figure 5-33).

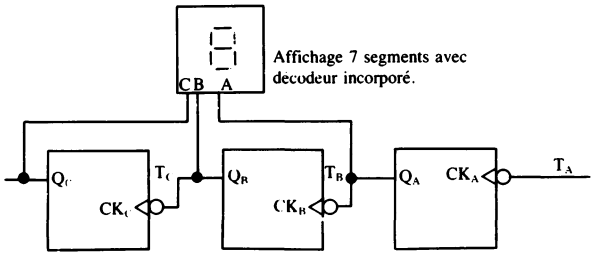


Figure 5-33 Compteur octal et affichage.

Pour l'affichage décimal, on n'utilisera que les trois premières entrées.

On peut donc réaliser sur ce principe des compteurs à base 2, 4, 8, 16, . . . , 2^n.

Ce montage en cascade ne peut pas être utilisé à n'importe quelle fréquence, si on veut décoder l'état à la sortie à un instant donné. Appelons t_p le *temps de propagation* dans une bascule, c'est-à-dire l'intervalle de temps entre la commande et l'apparition des données Q et $\bar{Q}$ à la sortie, le temps de propagation total entre le premier et le dernier étage du compteur sera de n t_p pour un compteur de n bascules. À ce temps, il faut ajouter le *temps de décodage* t_d. La fréquence maximale d'utilisation de ce compteur est donc

$$f_{max} = \frac{1}{nt_p + t_d}$$

Soit, par exemple, $t_p = 50$ ns et $t_d = 100$ ns. Pour un compteur modulo 32 (2^5), cinq bascules en cascade, la fréquence maximale d'utilisation sera

$$f_{max} = \frac{10^9}{5\,(50) + 100} = \frac{10^9}{350} = 2{,}8 \text{ MHz}$$

5-7-2 COMPTEURS ASYNCHRONES

Si on veut construire un compteur modulo N c'est-à-dire comptant jusqu'à N-1 et revenant à zéro, on cherche la puissance de 2 immédiatement supérieure à N. Soit, par exemple, un compteur modulo 10 ou compteur décimal, comptant jusqu'à 9 et repassant donc à zéro à la dixième impulsion d'horloge. La puissance de 2 immédiatement supérieure à 10 est $2^4 = 16$ (puisque $2^3 = 8$). L'exposant de cette puissance de 2 donne le nombre de bascules à utiliser, quatre dans notre exemple.

On câble ces bascules en compteur progressif (voir la figure 5-34), on connecte ensuite toutes les sorties Q qui sont à 1 pour N-1 à l'entrée d'une porte NON-ET et on ajoute une entrée d'horloge à cette porte NON-ET. La sortie de cette porte va aux entrées PRESET (mise à 1) des bascules restantes.

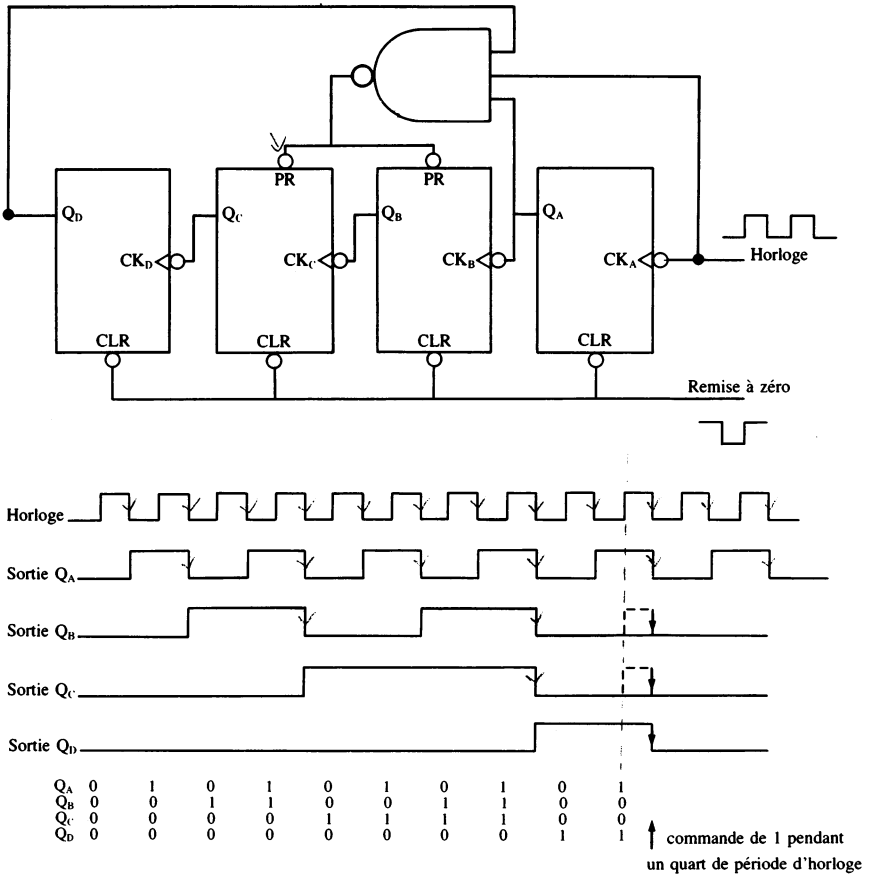


Figure 5-34 Compteur décimal progressif à entrées PRESET et chronogramme

Le fonctionnement de ce circuit est le suivant: tant que 1001 (9) n'est pas détecté, la sortie de la porte NON-ET est à 1 et n'exerce aucune influence sur Q_B et Q_C. Lorsqu'on détecte 1001, et que le signal d'horloge est 1, alors la sortie de la porte vaut 0 et commande prioritairement 1 aux sorties Q_B et Q_C (1111). Le compteur bascule à 0000 sur le front descendant d'impulsion d'horloge.

On obtient donc bien un compteur décimal à dix états et comptant de 0 à 9 en binaire naturel.

Une autre méthode peut être utilisée si les bascules disponibles n'ont pas d'entrée PRESET. Dans ce cas, on relie les sorties Q = 1 pour N aux entrées d'une porte NON-ET dont la sortie est la ligne CLEAR (remise à zéro, CLR) générale. La figure 5-35 représente un tel circuit.

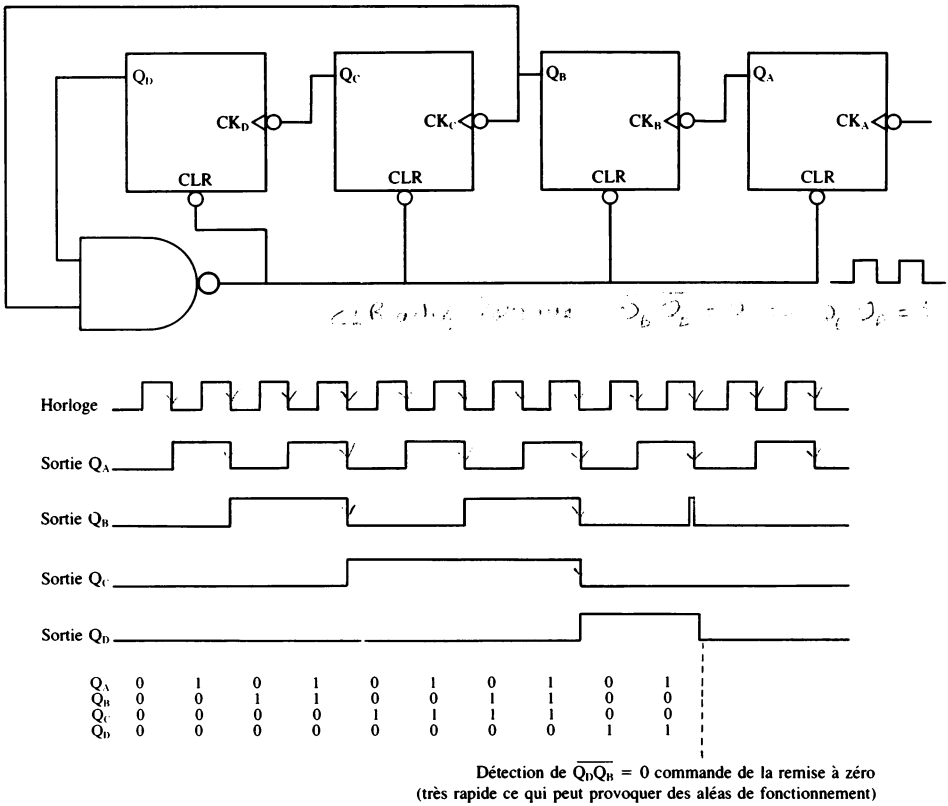


Figure 5-35 Compteur progressif décimal sans entrées PRESET et chronogramme

À la dixième impulsion d'horloge, la sortie de la porte NON-ET devient 0 et remet à zéro tout le compteur. Cette méthode est plus simple que la précédente, mais de réputation moins sûre. En effet, supposons que

le temps de propagation de la remise à zéro de la première bascule soit de 30 ns et celui de la bascule D de 10 ns, l'impulsion de remise à zéro, qui dure tant que $Q_D = 1$, aura donc une largeur de 10 ns. Elle peut donc être trop brève pour remettre à zéro la bascule A qui, dès lors, restera à 1. Le compteur repartira donc de 1 au lieu de 0. On aura une erreur d'une impulsion. Si c'était l'inverse, au point de vue temps de propagation, le compteur oscillerait entre 8 et 9. Des temps de propagation aussi différents se rencontrent lorsque les charges des bascules sont différentes.

Nous venons de voir une façon systématique de construire des compteurs asynchrones. Nous avons vu que ce montage en cascade avait un inconvénient. Il faut attendre que chaque impulsion d'horloge se propage d'un bout à l'autre du compteur. Cela limite la fréquence maximale d'utilisation surtout pour les cycles de comptage assez longs. On peut contourner cette difficulté, au prix d'une plus grande complexité du circuit, en commandant toutes les bascules en même temps. La fréquence maximale d'utilisation ne sera alors limitée que par la bascule la plus lente. Ce type de compteur s'appelle un compteur *synchrone*.

5-7-3 COMPTEURS SYNCHRONES

Dans ce cas on doit commander par le signal d'horloge toutes les bascules en même temps. Les états des bascules JK, par exemple, dépendent des entrées J et K. Pour un compteur modulo 2^3, par exemple, nécessitant donc trois bascules JK (C, B, A) on devra avoir la table de fonctions:

	C	B	A
→	0	0	0
	0	0	1
	0	1	0
	0	1	1
	1	0	0
	1	0	1
	1	1	0
	1	1	1
	0	0	0

La première bascule change donc d'état à chaque impulsion d'horloge. Ceci s'obtient en faisant J = K = 1 et à l'aide d'un signal d'horloge. La bascule B change d'état toutes les deux impulsions d'horloge. Son état dépend de A. L'état de la bascule C dépend de A et B. Pour résoudre ce genre de problème dressons tout d'abord la table des états recherchés.

Q_C \ Q_BQ_A	00	01	11	10
0	0	1	3	2
1	4	5	7	6

Q_A, Q_B, Q_C: sorties du compteur octal.

Figure 5-36 Table des différents états du compteur

Indiquons ensuite dans les tables l'état commandé à la bascule. Cet état dépend de J et K, nous dressons donc deux tables par bascule, une pour J, l'autre pour K. Pour la bascule A, on connaît déjà la solution $J_A = K_A = 1$. Exposons la façon de procéder, y compris pour la bascule A.

Pour mémoire, rappelons la table d'excitation de la bascule JK étudiée précédemment (voir plus haut).

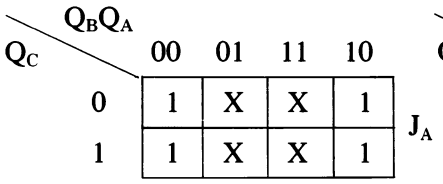
Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

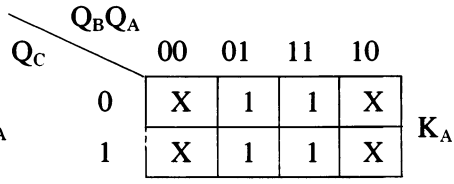
À l'état (0) la sortie Q_A est 0. À l'impulsion d'horloge suivante la sortie Q_A doit être égale à 1, il faut donc commander J = 1 et K = X. Écrivons ces valeurs dans la case de l'état (0) pour arriver à l'état (1) avec $Q_A = 1$ à l'impulsion d'horloge suivante. À l'état (1) Q_A vaut 1, il faut commander 0 donc il faut J = X et K = 1. Écrivons ces valeurs dans la case de l'état (1).

Dans les cases de l'état (2) écrivons J = 1 et K = X
Dans les cases de l'état (3) écrivons J = X et K = 1
Dans les cases de l'état (4) écrivons J = 1 et K = X
Dans les cases de l'état (5) écrivons J = X et K = 1
Dans les cases de l'état (6) écrivons J = 1 et K = X
Dans les cases de l'état (7) écrivons J = X et K = 1

nous revenons de cette façon à l'état 0.

Si on considère X comme des 1, on obtient les commandes $J_A = 1$ et $K_A = 1$.

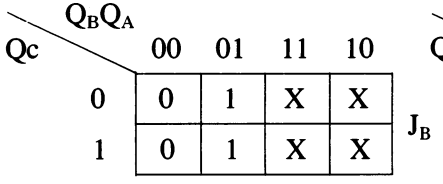


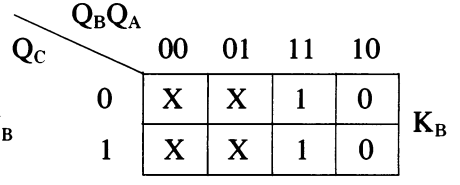


$J_A = 1$ $K_A = 1$

Procédons de la même façon pour la bascule B.

Dans les cases de l'état (0) écrivons J = 0 et K = X
Dans les cases de l'état (1) écrivons J = 1 et K = X
Dans les cases de l'état (2) écrivons J = X et K = 0
Dans les cases de l'état (3) écrivons J = X et K = 1
Dans les cases de l'état (4) écrivons J = 0 et K = X
Dans les cases de l'état (5) écrivons J = 1 et K = X
Dans les cases de l'état (6) écrivons J = X et K = 0
Dans les cases de l'état (7) écrivons J = J et K = 1

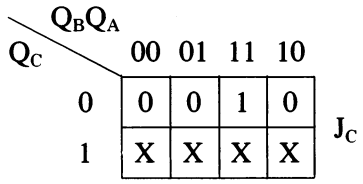


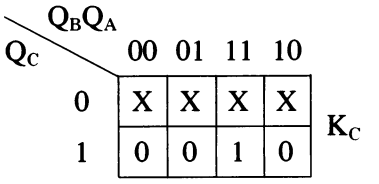


$J_B = Q_A$ $K_B = Q_A$

d'où les équations $J_B = Q_A = K_B$ déduites des tables de Karnaugh.

Pour la bascule C on aura les tables de Karnaugh suivantes:





$J_C = K_C = Q_A = Q_B$

D'où le schéma du compteur:

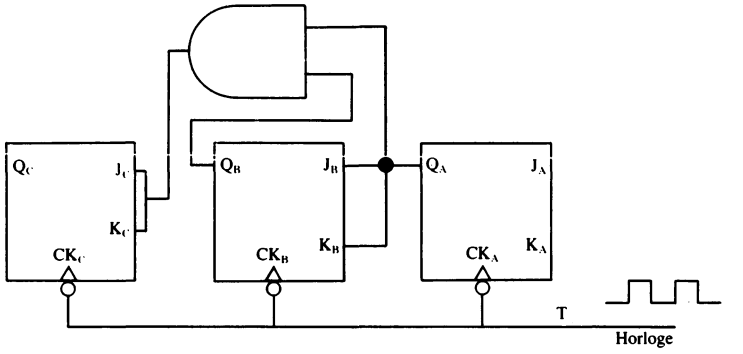


Figure 5-37 Compteur synchrone modulo 8

Pour ce type de compteur, il est intéressant d'utiliser des bascules en circuits intégrés avec des portes ET sur les entrées J et K. On n'a pas besoin d'ajouter des portes supplémentaires (voir les circuits 7470; 7472; 74102; 74110).

5-7-4 COMPTEUR SYNCHRONE DÉCIMAL

Dans ce cas on a besoin de quatre bascules. La table des états apparaît à la figure 5-38. On retient dix états sur les seize possibles. Les états à rejeter sont indiqués par des X. Après l'état 9 le compteur revient à l'état 0 .

Q_DQ_C \ Q_BQ_A	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	X	X	X	X
10	8	9	X	X

Figure 5-38 Table des états d'un compteur décimal synchrone

De cette table des états, déduisons les *tables de commande* des quatre bascules. On obtient:

Q_DQ_C \ Q_BQ_A	00	01	11	10
00	1	X	X	1
01	1	X	X	1
11	X	X	X	X
10	1	X	X	X

J_A

$J_A = 1$

Q_DQ_C \ Q_BQ_A	00	01	11	10
00	X	1	1	X
01	X	1	1	X
11	X	X	X	X
10	X	1	X	X

K_A

$K_A = 1$

Q_DQ_C \ Q_BQ_A	00	01	11	10
00	0	1	X	X
01	0	1	X	X
11	X	X	X	X
10	0	0	X	X

J_B

$J_B = Q_A\overline{Q_D}$

Q_DQ_C \ Q_BQ_A	00	01	11	10
00	X	X	1	0
01	X	X	1	0
11	X	X	X	X
10	X	X	X	X

K_B

$K_B = Q_A$

Q_DQ_C \ Q_BQ_A	00	01	11	10
00	0	0	1	0
01	X	X	X	X
11	X	X	X	X
10	0	0	X	X

J_C

$J_C = Q_AQ_B$

Q_DQ_C \ Q_BQ_A	00	01	11	10
00	X	X	X	X
01	0	0	1	0
11	X	X	X	X
10	X	X	X	X

K_C

$K_C = Q_AQ_B$

Q_DQ_C \ Q_BQ_A	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	X	X	X	X
10	X	X	X	X

J_D

$J_D = Q_AQ_BQ_C$

Q_DQ_C \ Q_BQ_A	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	X	X	X	X
10	0	1	X	X

K_D

$K_D = Q_A$

D'où l'implantation illustrée à la figure 5-39.

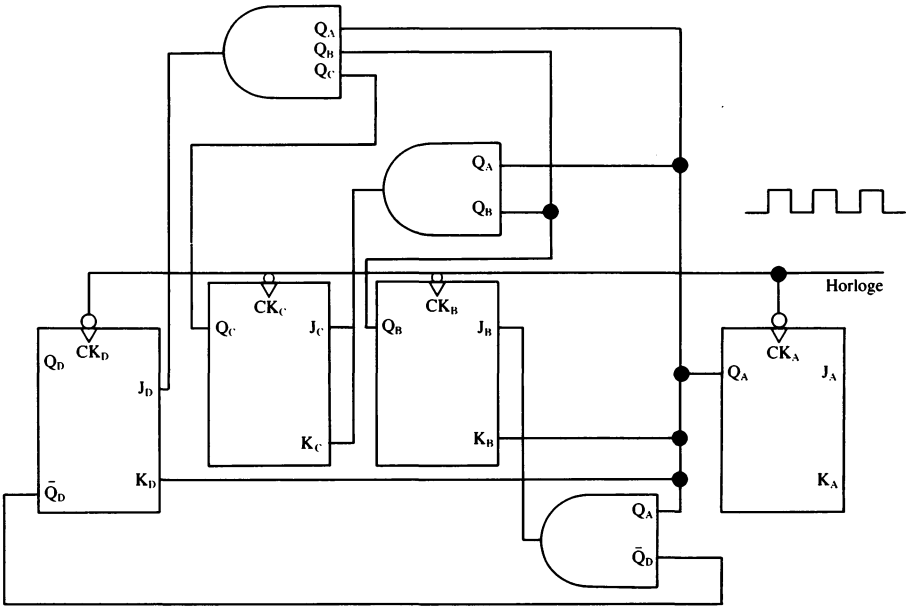


Figure 5-39 Compteur décimal synchrone

On utilise trois portes ET. Si on choisit des circuits intégrés genre 7472 on obtient le schéma de câblage plus simple de la figure 5-40.

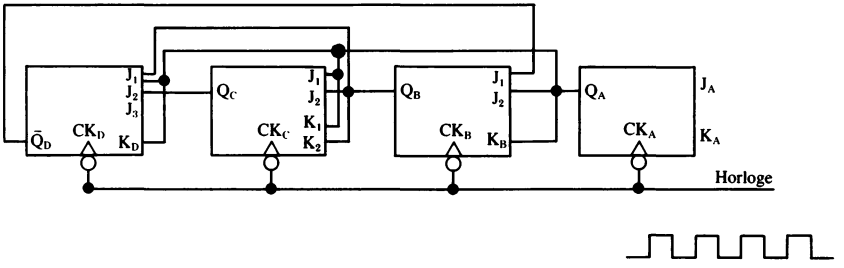


Figure 5-40 Compteur décimal synchrone (bascules JK 7472)

Les deux compteurs traités dans cette section sont des exemples typiques de circuits séquentiels. Nous avons vu une méthode générale de conception de compteurs. La résolution des exercices en fin de chapitre vous rendra cette méthode familière.

5-7-5 QUELQUES COMPTEURS SOUS FORME DE CIRCUITS INTÉGRÉS

Les compteurs 7490 (modulo 10), 7492 (modulo 12) et 7493 (modulo 16) sont des compteurs asynchrones composés de quatre bascules dont les connexions internes varient selon le type de compteurs. Chacun de ces compteurs a deux entrées (INPUT A et INPUT B) ce qui facilite leur utilisation. Ils comprennent chacun une bascule (diviseur par 2) qui peut être commandée directement et trois bascules câblées de façon à obtenir

un diviseur par 5 — 2 X 5 = compteur décimal,
un diviseru par 6 — 2 X 6 = compteur modulo 12,
ou un diviseur par 8 — 2 X 8 = compteur hexadécimal.

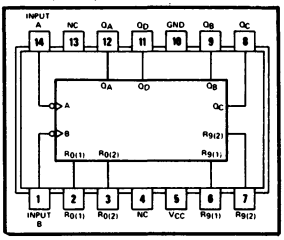
Remarquons que le compteur 7492 (modulo 12) ne compte pas dans l'ordre binaire naturel à partir de l'état 6. Le circuit 7490 peut se remettre à 0 ou à 9 selon la valeur des entrées R_0 et R_9.

7490A, 'L90, 'LS90 . . . DECADE COUNTERS

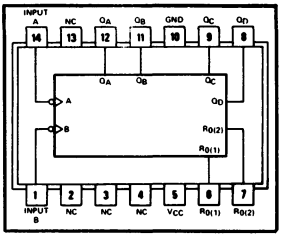
7492A, 'L92 . . . DIVIDE-BY-TWELVE COUNTERS

7493A, 'L93, 'LS93 . . . 4-BIT BINARY COUNTERS

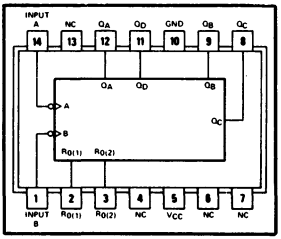
7490A, 'L90, 'LS90



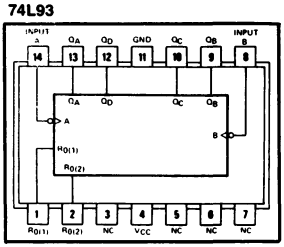
7492A, 'LS92



7493A, 'LS93

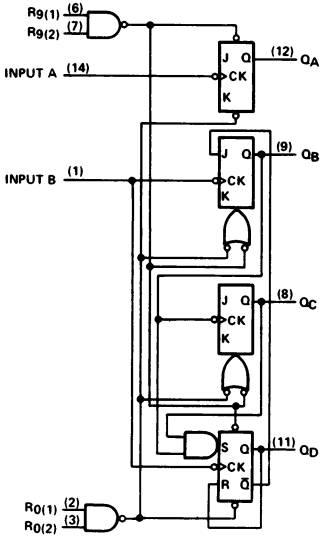


74L93

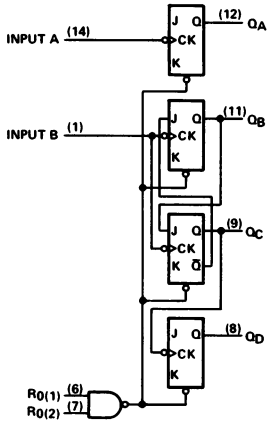


functional block diagrams

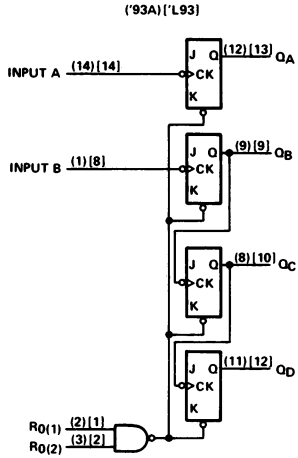
'90A, 'L90, 'LS90



'92A, 'LS92



'93A, 'L93, 'LS93



'90A, 'L90, 'LS90
BCD COUNT SEQUENCE
(See Note A)

COUNT	OUTPUT			
	Q_D	Q_C	Q_B	Q_A
0	L	L	L	L
1	L	L	L	H
2	L	L	H	L
3	L	L	H	H
4	L	H	L	L
5	L	H	L	H
6	L	H	H	L
7	L	H	H	H
8	H	L	L	L
9	H	L	L	H

'90A, 'L90, 'LS90
BI-QUINARY (5-2)
(See Note B)

COUNT	OUTPUT			
	Q_A	Q_D	Q_C	Q_B
0	L	L	L	L
1	L	L	L	H
2	L	L	H	L
3	L	L	H	H
4	L	H	L	L
5	H	L	L	L
6	H	L	L	H
7	H	L	H	L
8	H	L	H	H
9	H	H	L	L

'92A, 'LS92
COUNT SEQUENCE
(See Note C)

COUNT	OUTPUT			
	Q_D	Q_C	Q_B	Q_A
0	L	L	L	L
1	L	L	L	H
2	L	L	H	L
3	L	L	H	H
4	L	H	L	L
5	L	H	L	H
6	H	L	L	L
7	H	L	L	H
8	H	L	H	L
9	H	L	H	H
10	H	H	L	L
11	H	H	L	H

'93A, 'L93, 'LS93
COUNT SEQUENCE
(See Note C)

COUNT	OUTPUT			
	Q_D	Q_C	Q_B	Q_A
0	L	L	L	L
1	L	L	L	H
2	L	L	H	L
3	L	L	H	H
4	L	H	L	L
5	L	H	L	H
6	L	H	H	L
7	L	H	H	H
8	H	L	L	L
9	H	L	L	H
10	H	L	H	L
11	H	L	H	H
12	H	H	L	L
13	H	H	L	H
14	H	H	H	L
15	H	H	H	H

'90A, 'L90, 'LS90
RESET/COUNT FUNCTION TABLE

RESET INPUTS				OUTPUT			
$R_{0(1)}$	$R_{0(2)}$	$R_{9(1)}$	$R_{9(2)}$	Q_D	Q_C	Q_B	Q_A
H	H	L	X	L	L	L	L
H	H	X	L	L	L	L	L
X	X	H	H	H	L	L	H
X	L	X	L	COUNT			
L	X	L	X	COUNT			
L	X	X	L	COUNT			
X	L	L	X	COUNT			

'92A, 'LS92, '93A, 'L93, 'LS93
RESET/COUNT FUNCTION TABLE

RESET INPUTS		OUTPUT			
$R_{0(1)}$	$R_{0(2)}$	Q_D	Q_C	Q_B	Q_A
H	H	L	L	L	L
L	X	COUNT			
X	L	COUNT			

NOTES: A. Output Q_A is connected to input B for BCD count.
B. Output Q_D is connected to input A for bi-quinary count.
C. Output Q_A is connected to input B.
D. H = high level, L = low level, X = irrelevant

Parmi les compteurs synchrones en circuits intégrés citons la série 74160 à 74163.

Les compteurs 74160 et 74162 sont de modulo 10, leur remise à zéro (CLEAR) est asynchrone pour le premier et synchrone pour le second. Le 74161 et le 74163 sont des compteurs modulo 16, leur remise à zéro (CLEAR) est asynchrone pour le premier et synchrone pour le second.

Tous ces compteurs sont programmables, c'est-à-dire qu'on peut les mettre à l'état que l'on veut à l'aide des entrées ABCD et en mettant les broches 7 et 10 (ENABLE P et T) à 0. Dans ce cas, le compteur ne fonctionne pas mais met en mémoire la valeur des entrées ABCD. Dès que les broches 7 et 10 sont à 1, le compteur part de l'état mis en mémoire.

Une sortie, broche 15: RIPPLE CARRY OUTPUT *(Sortie du report en cascade)* permet de mettre ces compteurs en cascade et de les utiliser à hautes fréquences, car le signal de départ de l'autre compteur n'est pas donné par Q_D, on diminue le retard de propagation (voir le schéma).

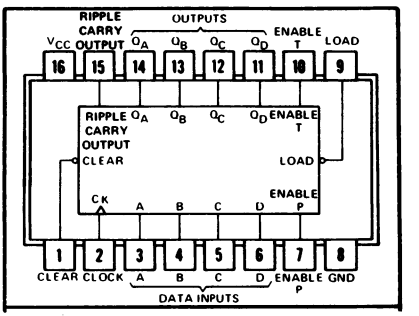
Ces caractéristiques illustrent les possiblités des circuits intégrés à grande échelle,

Les applications des compteurs sont innombrables. Ils permettent par exemple de mesurer la fréquence ou nombre d'impulsions pendant une seconde.

Pour concevoir une *horloge numérique*, par exemple, on transforme la tension du secteur 60 Hz en une tension de crête de 5 V que l'on redresse en éliminant le cycle négatif (redresseur demi-onde). Le comptage jusqu'à 60 (compteur modulo 6 suivi d'un modulo 10) donne les secondes. Un deuxième comptage de même style donne les minutes et un troisième les heures à l'aide d'un compteur modulo 12.

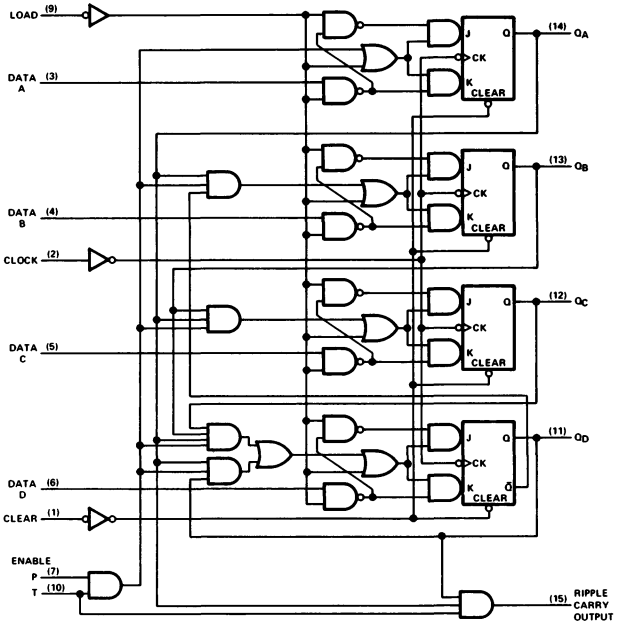
74160, '161, 'LS160A, 'LS161A... SYNCHRONOUS COUNTERS WITH DIRECT CLEAR
74162, '163, 'LS162A, 'LS163A, 'S162, 'S163... FULLY SYNCHRONOUS COUNTERS

SERIES 54', 54LS', 54S'...
SERIES 74', 74LS', 74S'...



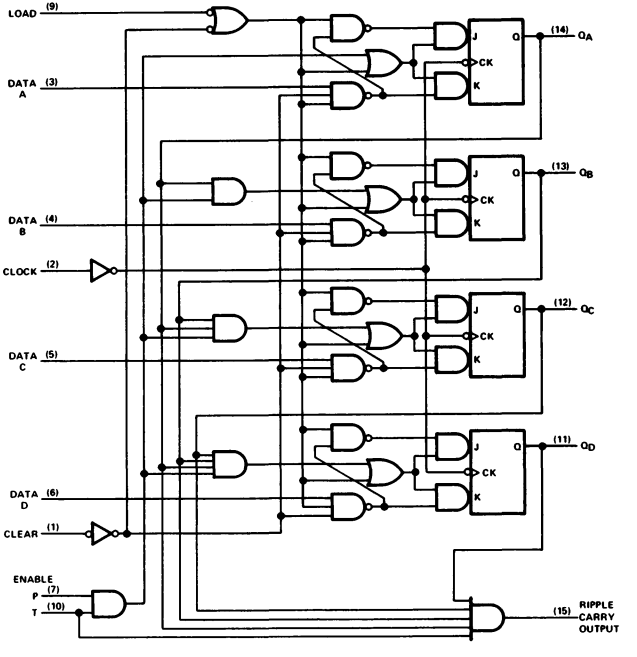
SN54160, SN74160 SYNCHRONOUS DECADE COUNTERS

SN54162, SN74162 synchronous decade counters are similar; however the clear is synchronous as shown for the SN54163, SN74163 binary counters at right.



SN54163, SN74163 SYNCHRONOUS BINARY COUNTERS

SN54161, SN74161 synchronous binary counters are similar; however, the clear is asynchronous as shown for the SN54160, SN74160 decade counters ar left.

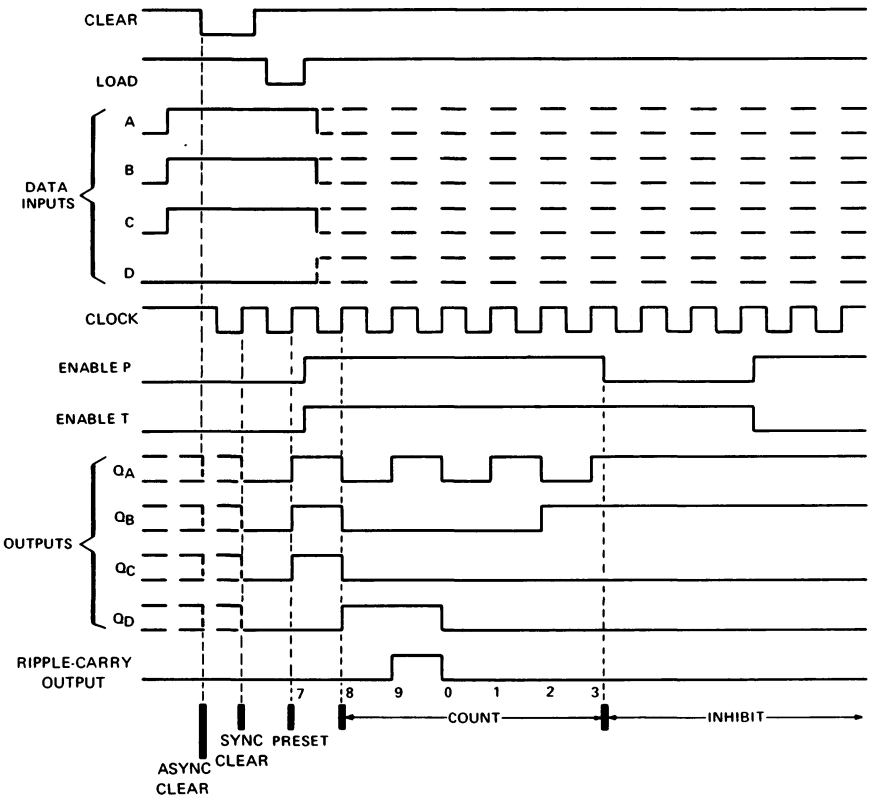


'160, '162, 'LS160A, 'LS162A, 'S162 DECADE COUNTERS

typical clear, preset, count, and inhibit sequences

Illustrated below is the following sequence:

1. Clear outputs to zero ('160 and 'LS160A are asynchronous; '162, 'LS162A,and 'S162 are synchronous)
2. Preset to BCD seven
3. Count to eight, nine, zero, one, two, and three
4. Inhibit

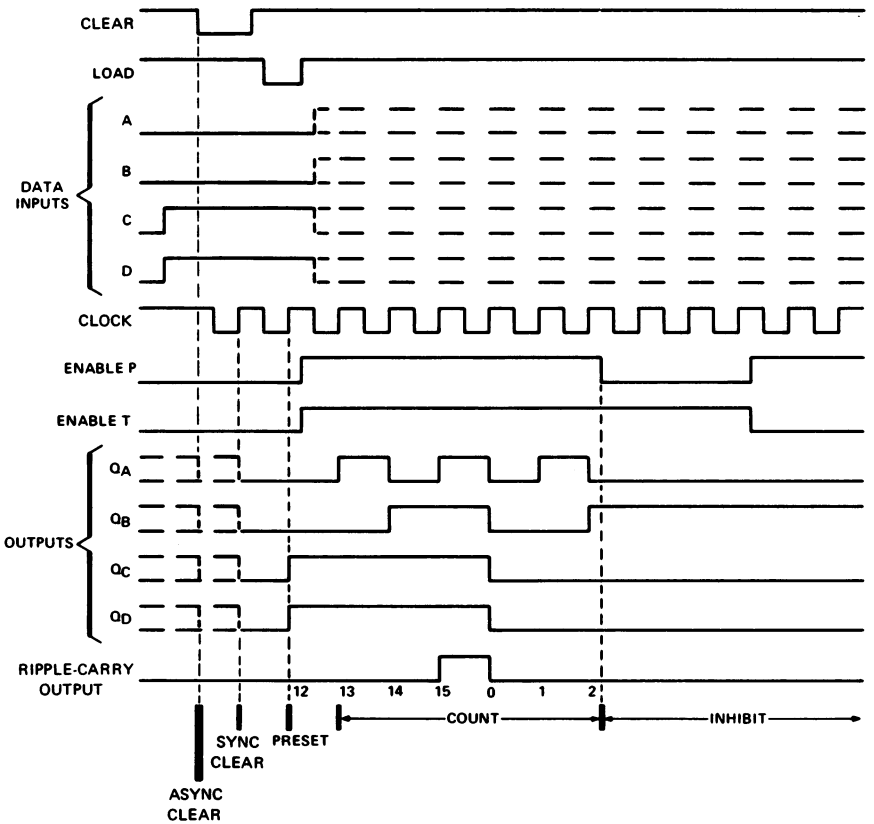


'161, 'LS161A, '163, 'LS163A, 'S163 BINARY COUNTERS

typical clear, preset, count, and inhibit sequences

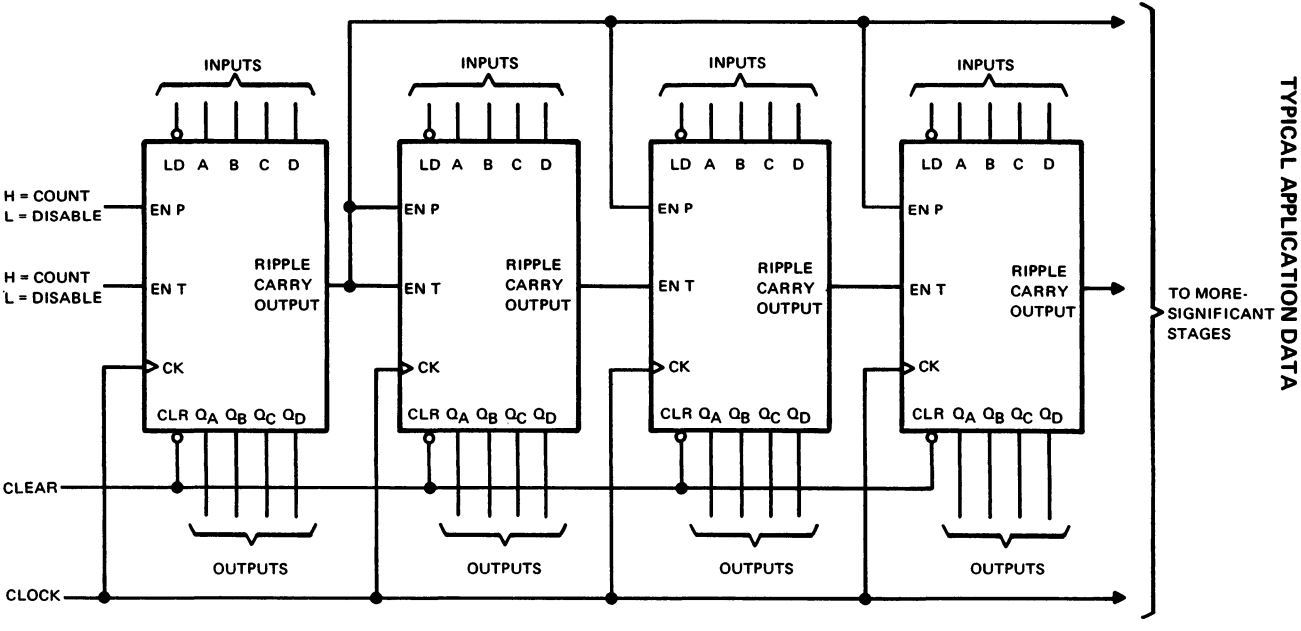
Illustrated below is the following sequence:

1. Clear outputs to zero ('161 and 'LS161A are asynchronous; '163, 'LS163A, and 'S163 are synchronous)
2. Preset to binary twelve
3. Count to thirteen, fourteen fifteen, zero, one, and two
4. Inhibit



N-BIT SYNCHRONOUS COUNTERS

This application demonstrates how the look-ahead carry circuit can be used to implement a high-speed n-bit counter. The '160, '162, 'LS160A, 'LS162A, or 'S162 will count in BCD and the '161, '163, 'LS161A, 'LS163A or 'S163 will count in binary. Virtually any count mode (modulo-N, N_1-to-N_2, N_1-to-maximum) can be used with this fast look-ahead circuit.



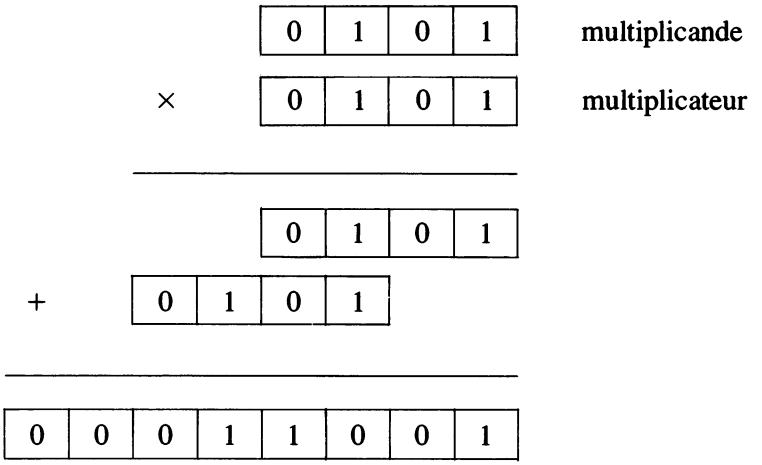
TYPICAL APPLICATION DATA

5-8 REGISTRES À DÉCALAGE

5-8-1 PRINCIPE DES REGISTRES À DÉCALAGE

Dans certains cas, comme par exemple la multiplication en binaire, il est nécessaire de décaler un nombre à gauche ou à droite.

Considérons la multiplication suivante.



Mettons le multiplicateur et le multiplicande dans un registre.

Regardons le bit de droite du multiplicateur. S'il est égal à 1, additionner le multiplicande; s'il est égal à 0 additionner zéro. Décalons le multiplicande d'un rang vers la gauche (ou le multiplicateur d'un rang vers la droite) et regardons le bit suivant du multiplicateur et appliquons la règle: s'il est égal à 1, additionner le multiplicande, s'il est égal à 0 additionner 0. Répétons ce processus jusqu'à considération de tous les bits du multiplicateur. Si les registres sont constitués de bascules JK ayant en mémoire les nombres à multiplier, il faut un circuit capable de décaler vers la gauche ou vers la droite les nombres de ce registre.

Soit, par exemple le registre de quatre bits contenant:

0	1	1	0

Il faut qu'à la première commande de décalage *à gauche* on obtienne:

1	1	0	0

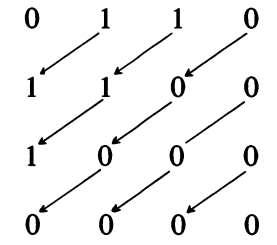
à la deuxième

1	0	0	0

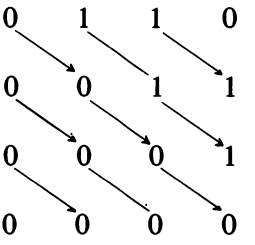
à la troisième

0	0	0	0

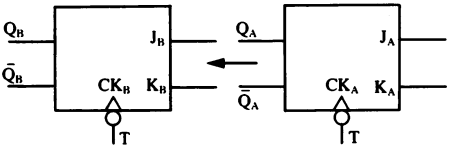
soit la séquence



Pour un décalage *à droite* on aura la séquence:



Le problème consiste à donc à trouver un circuit qui transfère la valeur d'une bascule à une autre.



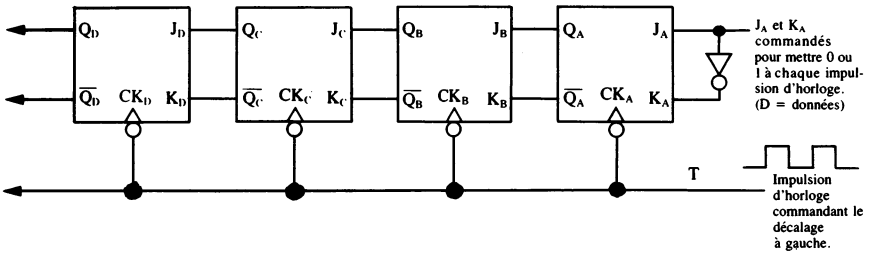
Il faut $Q_B = Q_A$ après l'ordre de décalage.

Si $Q_A = 1$, il faut $Q_B = 1$ donc commander $J_B = 1$ et $K_B = 0$.

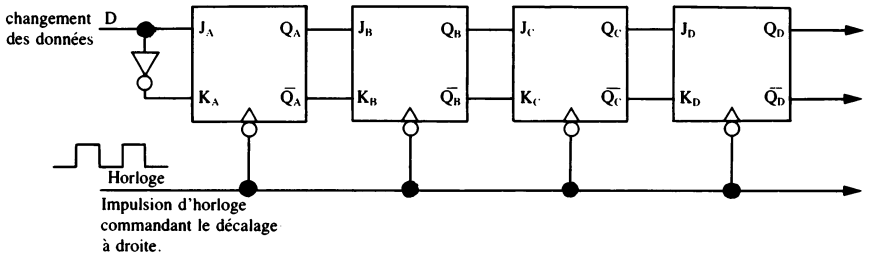
Si $Q_A = 0$, il faut $Q_B = 0$ donc commander $J_B = 0$ et $K_B = 1$.

On peut donc écrire immédiatement: $J_B = Q_A$ et $K_B = \overline{Q_A}$, $J_C = Q_B$ et $K_C = \overline{Q_B}$ etc.

On obtient le schéma de registre à décalage de quatre bits suivant:



Pour un décalage à droite, on a le schéma symétrique:



Pour entrer des nombres en *série* dans ces registres, il suffit donc de brancher la bascule A en bascule D, d'entrer chaque bit et de décaler.

On peut aussi entrer le nombre en *parallèle* en se servant des entrées asynchrones PRESET et CLEAR.

Un circuit combinatoire plus complexe permettrait d'entrer le nombre en commandant les entrées J et K de chaque bascule.

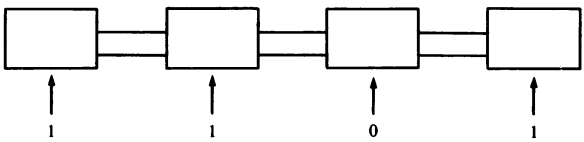
5-8-2 QUELQUES REGISTRES À DÉCALAGE EN CIRCUITS INTÉGRÉS

Parmi les caractéristiques des registres à décalage citons:

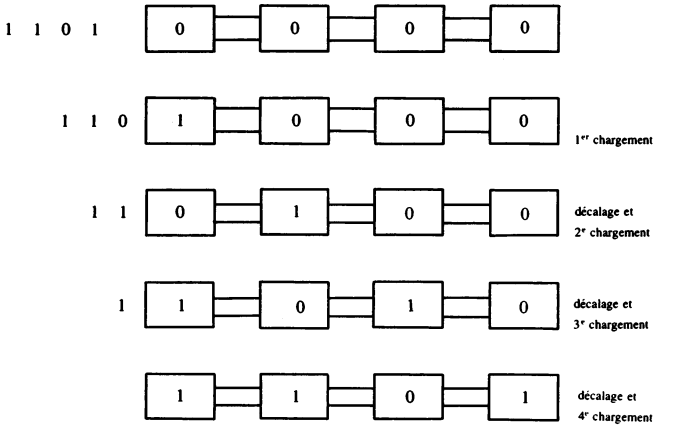
1) Les trois types de décalage: à droite, à gauche ou dans ces deux directions par une commande de mode (registres *bidirectionnels*).

2) Les deux types de chargements des registres: parallèle ou série. En parallèle, on entre le nombre dans les bascules d'un seul coup. En série, on entre le premier bit du nombre dans la première bascule, de gauche ou de droite, on le décale en même temps que l'on entre le deuxième bit, etc.

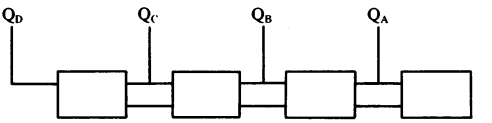
La figure ci-dessous illustre le chargement parallèle du nombre 1101.



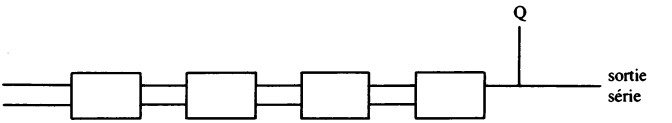
Pour le chargement série de ce même nombre on aura un premier chargement et une suite de décalages et chargements selon la figure suivante:



3) Les deux types de sorties des registres, parallèle ou série. La figure ci-dessous illustre une sortie parallèle.



Chaque bascule a la sortie Q ou les deux sorties Q et $\overline{Q}$ accessibles. Le nombre sort d'un seul coup. La figure ci-dessous illustre une sortie série. Dans ce cas, les bits sont accessibles un par un, après décalage:



Toutes les combinaisons des caractéristiques 2) et 3) sont possibles, soit:

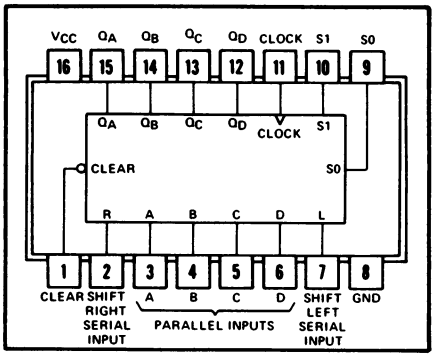
— entrée parallèle, sortie parallèle
— entrée parallèle, sortie série
— entrée série, sortie parallèle
— entrée série, sortie série.

4) Le nombre de bits dans les registres est de quatre, cinq ou huit. Un registre composé de ces registres élémentaires peut contenir tout nombre de bits désiré.

Les registres à décalage sont fabriqués en circuits intégrés. Le 74194 est un registre à décalage de quatre bits à entrée et sortie parallèle ou entrée série à décalage à droite ou à gauche. Le 7496 est un registre de cinq bits à entrée et sortie parallèle ou entrée et sortie série, à décalage à droite. Le 74198 est un registre à décalage de huit bits à chargement parallèle ou série à décalage à droite ou à gauche et sortie parallèle.

Voir ci-dessous les planches relatives à ces registres à décalage.

SN74194, SN74LS194A, SN74S194 . . .



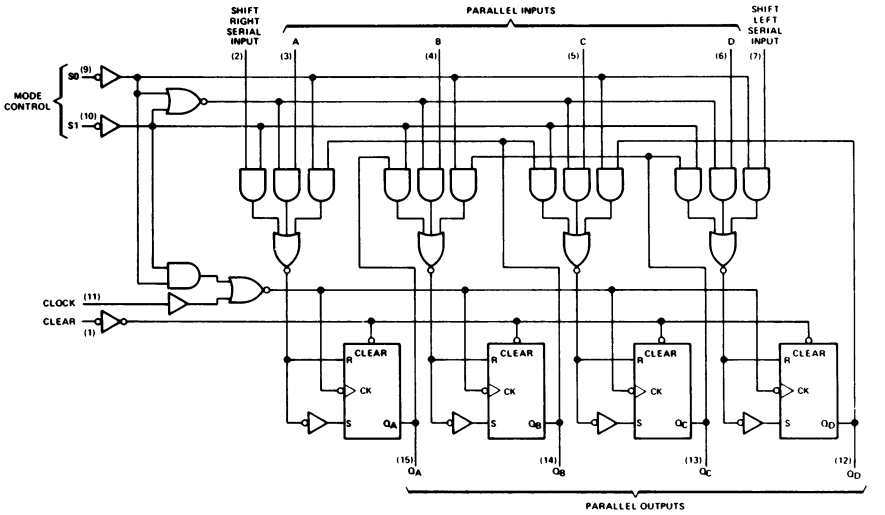
FUNCTION TABLE

INPUTS										OUTPUTS			
CLEAR	MODE		CLOCK	SERIAL		PARALLEL				Q_A	Q_B	Q_C	Q_D
	S1	S0		LEFT	RIGHT	A	B	C	D				
L	X	X	X	X	X	X	X	X	X	L	L	L	L
H	X	X	L	X	X	X	X	X	X	Q_{A0}	Q_{B0}	Q_{C0}	Q_{D0}
H	H	H	↑	X	X	a	b	c	d	a	b	c	d
H	L	H	↑	X	H	X	X	X	X	H	Q_{An}	Q_{Bn}	Q_{Cn}
H	L	H	↑	X	L	X	X	X	X	L	Q_{An}	Q_{Bn}	Q_{Cn}
H	H	L	↑	H	X	X	X	X	X	Q_{Bn}	Q_{Cn}	Q_{Dn}	H
H	H	L	↑	L	X	X	X	X	X	Q_{Bn}	Q_{Cn}	Q_{Dn}	L
H	L	L	X	X	X	X	X	X	X	Q_{A0}	Q_{B0}	Q_{C0}	Q_{D0}

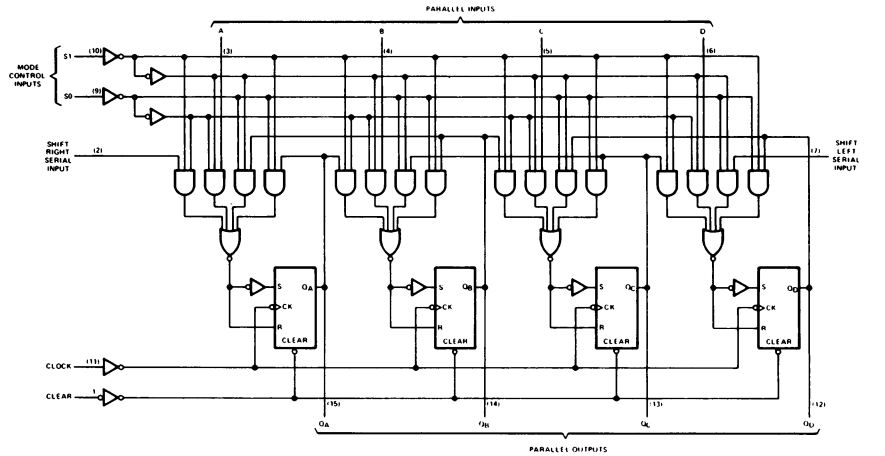
H = high level (steady state)
L = low level (steady state)
X = irrelevant (any input, including transitions)
↑ = transition from low to high level
a, b, c, d = the level of steady-state input at inputs A, B, C, or D, respectively.
Q_{A0}, Q_{B0}, Q_{C0}, Q_{D0} = the level of Q_A, Q_B, Q_C, or Q_D, respectively, before the indicated steady-state input conditions were established.
Q_{An}, Q_{Bn}, Q_{Cn}, Q_{Dn} = the level of Q_A, Q_B, Q_C, respectively, before the most-recent ↑ transition of the clock.

functional block diagrams

'194

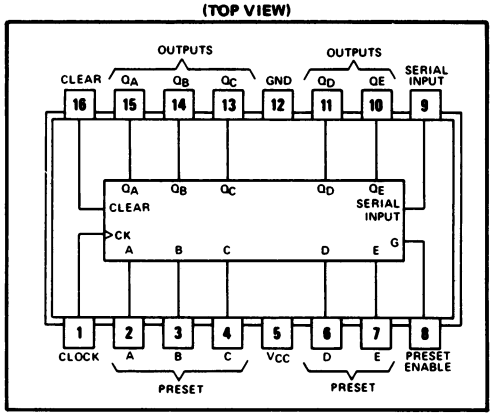


'LS194A, 'S194



SN7496, SN74L96, SN74LS96 . . .

(TOP VIEW)

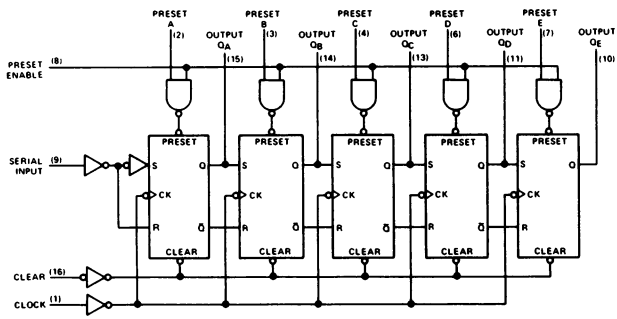


FUNCTION TABLE

INPUTS									OUTPUTS				
CLEAR	PRESET ENABLE	PRESET A	PRESET B	PRESET C	PRESET D	PRESET E	CLOCK	SERIAL	Q_A	Q_B	Q_C	Q_D	Q_E
L	L	X	X	X	X	X	X	X	L	L	L	L	L
L	X	L	L	L	L	L	X	X	L	L	L	L	L
H	H	H	H	H	H	H	X	X	H	H	H	H	H
H	H	L	L	L	L	L	L	X	Q_{A0}	Q_{B0}	Q_{C0}	Q_{D0}	Q_{E0}
H	H	H	L	H	L	H	L	X	H	Q_{B0}	H	Q_{D0}	H
H	L	X	X	X	X	X	L	X	Q_{A0}	Q_{B0}	Q_{C0}	Q_{D0}	Q_{E0}
H	L	X	X	X	X	X	↑	H	H	Q_{An}	Q_{Bn}	Q_{Cn}	Q_{Dn}
H	L	X	X	X	X	X	↑	L	L	Q_{An}	Q_{Bn}	Q_{Cn}	Q_{Dn}

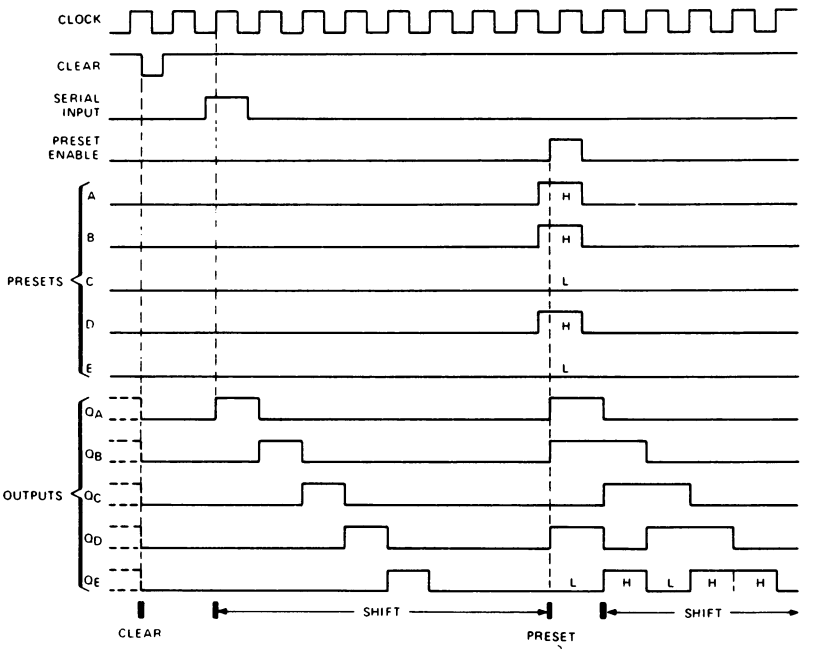
H = high level (steady state), L = low level (steady state)
X = irrelevant (any input, including transitions)
↑ = transition from low to high level
Q_{A0}, Q_{B0}, etc = the level of Q_A, Q_B, etc, respectively before the indicated steady-state input conditions were established.
Q_{An}, Q_{Bn}, etc = the level of Q_A, Q_B, etc, respectively before the most-recent ↑ transition of the clock.

functional block diagram

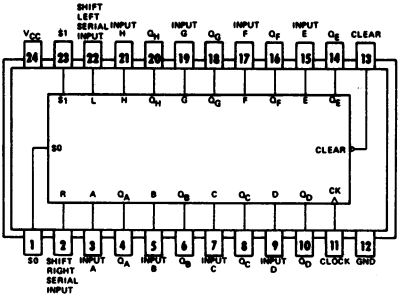


. . . dynamic input activated by transition from a high level to a low level.

typical clear, shift, preset, and shift sequences



SN74198 .



'198

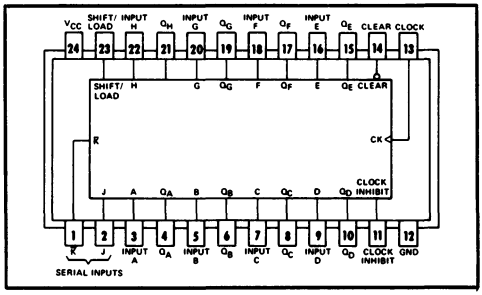
FUNCTION TABLE

INPUTS							OUTPUTS			
CLEAR	MODE		CLOCK	SERIAL		PARALLEL	Q_A	Q_B ...	Q_G	Q_H
	S_1	S_0		LEFT	RIGHT	A ... H				
L	X	X	X	X	X	X	L	L	L	L
H	X	X	L	X	X	X	Q_{A0}	Q_{B0}	Q_{G0}	Q_{H0}
H	H	H	↑	X	X	a ... h	a	b	g	h
H	L	H	↑	X	H	X	H	Q_{An}	Q_{Fn}	Q_{Gn}
H	L	H	↑	X	L	X	L	Q_{An}	Q_{Fn}	Q_{Gn}
H	H	L	↑	H	X	X	Q_{Bn}	Q_{Cn}	Q_{Hn}	H
H	H	L	↑	L	X	X	Q_{Bn}	Q_{Cn}	Q_{Hn}	L
H	L	L	X	X	X	X	Q_{A0}	Q_{B0}	Q_{G0}	Q_{H0}

H = high level (steady state), L = low level (steady state)
X = irrelevant (any input, including transitions)
↑ = transition from low to high level
a . . . h = the level of steady-state input at inputs A thru H, respectively.
Q_{A0}, Q_{B0}, Q_{G0}, Q_{H0} = the level of Q_A, Q_B, Q_G, or Q_H, respectively, before the indicated steady-state input conditions were established.
Q_{An}, Q_{Bn}, etc. = the level of Q_A, Q_B, etc., respectively, before the most-recent ↑ transition of the clock.

SN74199
8-BIT SHIFT REGISTERS

SN74199 . . .



'199

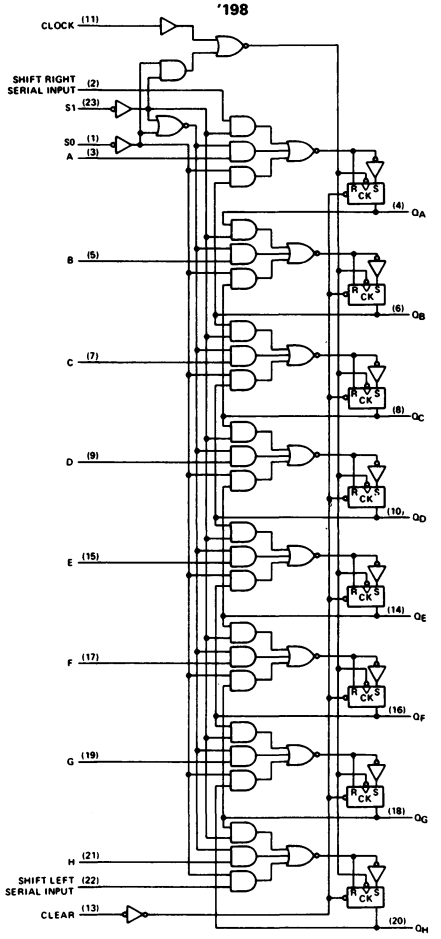
FUNCTION TABLE

INPUTS						OUTPUTS				
CLEAR	SHIFT/ LOAD	CLOCK INHIBIT	CLOCK	SERIAL		PARALLEL	Q_A	Q_B	Q_C ...	Q_H
				J	$\overline{K}$	A ... H				
L	X	X	X	X	X	X	L	L	L	L
H	X	L	L	X	X	X	Q_{A0}	Q_{B0}	Q_{C0}	Q_{H0}
H	L	L	↑	X	X	a ... h	a	b	c	h
H	H	L	↑	L	H	X	Q_{A0}	Q_{A0}	Q_{Bn}	Q_{Gn}
H	H	L	↑	L	L	X	L	Q_{An}	Q_{Bn}	Q_{Gn}
H	H	L	↑	H	H	X	H	Q_{An}	Q_{Bn}	Q_{Gn}
H	H	L	↑	H	L	X	$\overline{Q}_{An}$	Q_{An}	Q_{Bn}	Q_{Gn}
H	X	H	↑	X	X	X	Q_{A0}	Q_{B0}	Q_{B0}	Q_{H0}

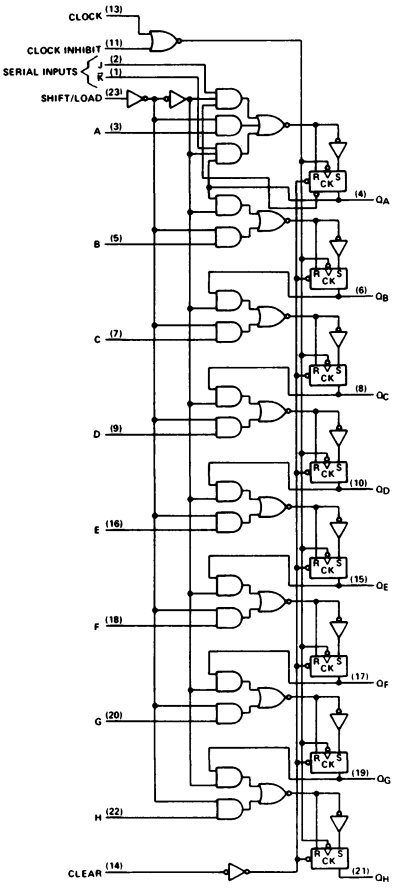
H = high level (steady state), L = low level (steady state)
X = irrelevant (any input, including transitions)
↑ = transition from low to high level
a . . . h = the level of steady-state input at inputs A thru H, respectively.
Q_{A0}, Q_{B0}, Q_{C0} . . . Q_{H0} = the level of Q_A, Q_B, or Q_C thru Q_H, respectively, before the indicated steady-state input conditions were established.
Q_{An}, Q_{Bn} . . . Q_{Gn} = the level of Q_A or Q_B thru Q_G, respectively, before the most-recent ↑ transition of the clock.

functional block diagrams

'198



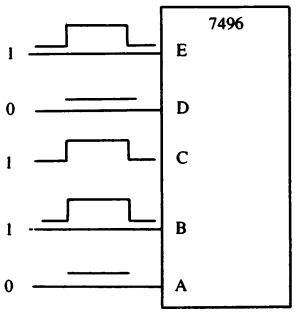
'199



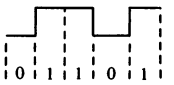
5-8-3 APPLICATIONS DES REGISTRES À DÉCALAGE

5-8-3-1 Conversion série-parallèle

Parmi les applications les plus simples des registres citons la conversion de nombres parallèle en nombre série. Considérons, par exemple, le registre à décalage de cinq bits 7496. On peut changer un nombre en parallèle dans ce registre, 01101 par exemple, selon la figure ci-dessous.



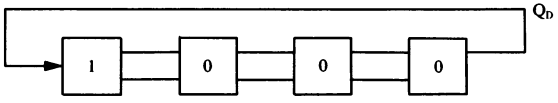
Sortons ce nombre à Q_E, à la cadence des impulsions de décalage.



Ce nombre, mis sous forme d'impulsions série est, par exemple, envoyé sur une ligne de transmission et reçu à l'entrée série (broche 9) d'un autre 7496. En synchronisant les horloges d'émission et de réception on peut récupérer ce nombre série EDCBA à la sortie parallèle.

5-8-3-2 Compteur en anneau

Considérons le registre à décalage de quatre bits 74194. Faisons les branchements de la ligne 5 de la table de fonctions après avoir chargé Q_A à 1 et branché *Shift right serial* input à Q_D (relier la broche 2 à la broche 12). Nous avons donc 1 0 0 0 dans le registre selon la configuration illustrée ci-dessous.



On y constate une rétroaction entre Q_D et l'entrée série. On a la suite des décalages à droite répétitive suivante.

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1
1	0	0	0

Un tel compteur est dit en anneau. Il comporte, dans ce cas, cinq états. Il présente l'avantage d'être très facile à décoder puisque pour chaque état une seule sortie est à 1.

PROBLÈMES

Faire le schéma d'un compteur asynchrone

5-1 Modulo 4

5-2 Modulo 5

5-3 Modulo 6

5-4 Modulo 7

5-5 Modulo 9

5-6 Modulo 11

5-7 Modulo 13

Faire le schéma d'un compteur synchrone à l'aide de bascules JK 7472 ou 7473

5-8 Modulo 3

5-9 Modulo 4

5-10 Modulo 5

5-11 Modulo 6

5-12 Modulo 7

5-13 Modulo 8

5-14 Modulo 9

5-15 Modulo 10

5-16 Modulo 11

5-17 Modulo 12

5-18 Modulo 13

5-19 Même question pour un compteur décimal de code plus trois.

5-20 Code Gray

5-21 Code décimal

2	4	2	1	(poids des bits)
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
1	0	1	1	5
1	1	0	0	6
1	1	0	1	7
1	1	1	0	8
1	1	1	1	9

où est disponible le complément à 9 de ces nombres

Implantation de compteurs

5-22 Connecter un circuit intégré 7490 en compteur modulo 6.

5-23 Connecter un 7492 en compteur modulo 7.

5-24 Connecter un 7492 en compteur modulo 9.

5-25 Connecter un 7492, ajouter une porte ET, en compteur modulo 11.

Connecter un 7493, ajouter une porte ET, en compteur:

5-26 Modulo 7

5-27 Modulo 11

5-28 Modulo 13

5-29 Modulo 14

5-30 Modulo 15

Logique séquentielle

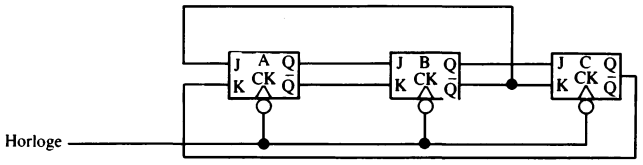
Connecter un 7493, sans ajouter de porte ET, en compteur:

5-31 Modulo 10

5-32 Modulo 9

5-33 Modulo 12

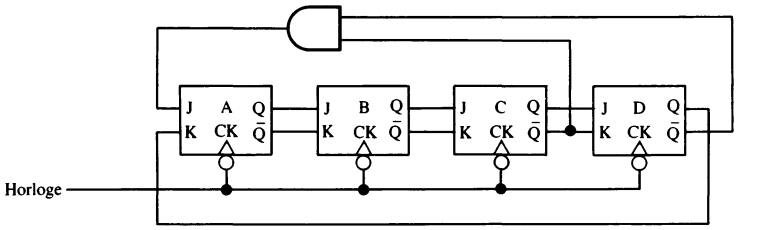
Considérer le compteur ci-dessous



5-34 Donner la séquence de ses états à partir de l'état 101.

5-35 Même question à partir de l'état 000.

Soit le compteur



5-36 Donner la séquence de ses états à partir de l'état 1001.

5-37 Même question à partir de l'état 1011.

Appendice A

AUTOMATISMES ET CAHIER DES CHARGES

Reproduit avec l'autorisation de l'Agence nationale pour le développement de la production automatisée (ADEPA)

Avant de décrire le GRAFCET, ses règles de fonctionnement et ses possibilités d'utilisation, il est nécessaire d'introduire une méthodologie dans la conception des systèmes automatisés.

Elle repose sur trois idées fondamentales :

- Dès la conception, le système à construire doit être décomposé en une ***partie opérative*** et une ***partie commande.*** Cette structure permet un dialogue profitable entre le futur utilisateur du système, et l'automaticien, responsable de la partie commande.
- Il importe de donner une description précise du fonctionnement de la partie commande, par une approche progressive des fonctions à remplir jusqu'à leur matérialisation.
- Le langage courant se prête mal à cette description. D'où la nécessité d'adopter un langage spécifique : le GRAFCET.

1.1 PARTIE OPÉRATIVE – PARTIE COMMANDE

D'une façon tout à fait générale, un système automatisé peut se décomposer en deux parties qui coopèrent : l'une est dite **partie opérative*** et l'autre **partie commande**.

Par exemple, dans une machine-outil à commande numérique, la partie opérative est la machine-outil proprement dite et la partie commande l'équipement de commande numérique.

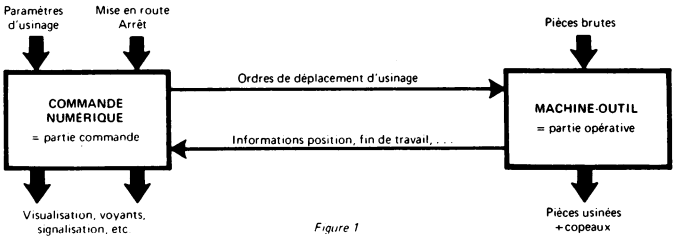


Figure 1

De même,dans un ascenseur,l'ensemble électro-mécanique (cabine, moteur, portes) constitue la partie opérative, les boutons d'appel, la logique et les armoires d'appareillage constituent la partie commande.

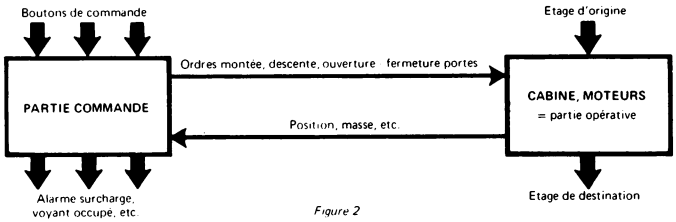


Figure 2

La partie opérative effectue des opérations (transformation de pièces brutes en pièces usinées, translation de la cabine de l'étage de départ à celui d'arrivée) lorsque l'ordre lui en est donné par la partie commande. Grâce aux comptes rendus (position, etc.) fournis par la partie opérative, la partie commande est tenue informée de l'état d'avancement des opérations effectuées.

Outre ce dialogue par ordres et comptes rendus avec la **partie opérative,** la partie **commande** échange des informations avec l'extérieur du système (pilote, usager, surveillant, . . .) dont elle reçoit des consignes et à qui elle fournit des comptes rendus visuels ou sonores.

Dans une machine-outil à commande numérique, la **partie commande** reçoit les paramètres d'usinage, les signaux de mise en marche ou d'arrêt, etc. allume les voyants, actionne les klaxons d'alarme . . . Dans un ascenseur, c'est grâce aux boutons à la disposition des usagers que la **partie commande** reçoit ses consignes, indique sur un synoptique l'étage où se trouve la cabine, le sens de son déplacement, actionne le voyant de surcharge, etc.

Pour résumer :

La **partie opérative** est le processus physique à automatiser. La **partie commande** est un automatisme qui élabore en sortie des ordres destinés au processus et des signaux de visualisation en fonction des comptes rendus venant du processus et des consignes qu'il reçoit en entrée. On se limitera ici aux automatismes logiques pour lesquels les informations traitées présentent un caractère "tout ou rien". Le cahier des charges d'un automatisme est la description de son comportement en fonction de l'évolution de son environnement, c'est-à-dire non seulement de ses entrées, mais aussi de ses conditions générales d'utilisation.

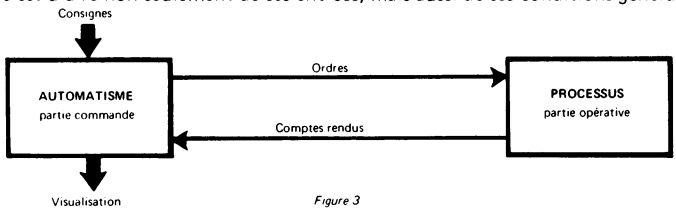


Figure 3

Nota : * La **partie opérative** est aussi appelée **partie puissance**
** La **partie commande** est aussi appelée **automate**

1.2 APPROCHE PROGRESSIVE DU CAHIER DES CHARGES DE LA PARTIE COMMANDE

L'automaticien chargé de la conception et de la réalisation de la partie commande doit rechercher dans le cahier des charges une description claire, précise, sans ambiguïtés ni omission, du rôle et des performances de l'équipement à réaliser.

Pour y parvenir, il est souhaitable de diviser la description en deux niveaux successifs et complémentaires :

- le premier niveau décrit le comportement de la partie commande vis-à-vis de la partie opérative : c'est le rôle des spécifications fonctionnelles permettant au concepteur de comprendre ce que l'automatisme doit faire, face aux différentes situations pouvant se présenter.
- le deuxième niveau ajoute aux exigences fonctionnelles les précisions indispensables aux conditions de fonctionnement des matériels, grâce aux spécifications technologiques et opérationnelles.

En sériant les problèmes, fonctionnels d'un côté, technologiques de l'autre, cette approche évite au lecteur de se sentir submergé d'emblée sous une foule de détails plus nuisibles qu'utiles.

1.2.1 Niveau 1 — Spécifications fonctionnelles

Les spécifications fonctionnelles caractérisent les réactions de l'automatisme face aux informations issues de la partie opérative, dans le but de faire comprendre au concepteur quel devra être le rôle de la partie commande à construire. Elles doivent donc définir de façon claire et précise les différentes fonctions, informations et commandes impliquées dans l'automatisation de la partie opérative, sans préjuger en aucune façon des technologies.

En conséquence, ni la nature ni les caractéristiques des différents capteurs ou actionneurs utilisés n'ont leur place dans ces spécifications. Peu importe, à ce niveau, que l'on effectue un déplacement à l'aide d'un vérin hydraulique ou pneumatique, ou encore d'un moteur électrique. Ce qu'il faut savoir c'est dans quelles circonstances ce déplacement doit s'effectuer.

Par contre, il importe que les sécurités de fonctionnement prévues soient incorporées dans les spécifications fonctionnelles, dans la mesure où elles ne dépendent pas directement de la technologie de ces capteurs ou actionneurs.

1.2.2 Niveau 2

Spécifications technologiques

Les spécifications technologiques précisent la façon dont l'automatisme devra physiquement s'insérer dans l'ensemble que constitue le système automatisé et son environnement. Ce sont les précisions à apporter en complément des spécifications fonctionnelles pour que l'on puisse concevoir un automatisme pilotant réellement la partie opérative.

C'est à ce niveau seulement que doivent intervenir les renseignements sur la nature exacte des capteurs et actionneurs employés, leurs caractéristiques et les contraintes qui peuvent en découler. A ces spécifications d'interface peuvent également s'ajouter des spécifications d'environnement de l'automatisme : température, humidité, poussières, anti-déflagrance, tensions d'alimentation, etc.

Spécifications opérationnelles

Les spécifications opérationnelles ont trait au suivi de fonctionnement de l'automatisme au cours de son existence. Il s'agit là des considérations concernant l'équipement une fois réalisé et mis en exploitation : fiabilité, absence de pannes dangereuses, disponibilité, possibilités de modification de l'équipement en fonction des transformations de la partie opérative, facilité de maintenance, dialogue homme - machine, etc.

Ces considérations, primordiales pour l'exploitant du processus à automatiser en raison de leurs répercussions sur le plan économique, sont souvent sous-estimées dans les cahiers des charges. Parfois difficiles à exprimer de façon quantitative, elles n'en ont pas moins d'incidence sur la manière de réaliser l'équipement.

1.3 NECESSITÉ D'UN OUTIL DE REPRÉSENTATION

Lorsque des spécifications sont exprimées en langage courant,il y a un risque permanent d'incompréhension ou de malentendu entre rédacteur et lecteur d'un cahier des charges.

En effet,certains mots sont peu précis, mal définis ou, ce qui est pire, possèdent plusieurs sens. Cela est particulièrement vrai pour les termes de jargon technique : parfaitement définis dans un certain contexte, ils pourront être pour un non-initié, soit totalement hermétiques, ce qui est un moindre mal, soit interprétés à contre-sens, ce qui est catastrophique.

Le langage courant se révèle en outre assez mal adapté à la description précise des systèmes séquentiels, en particulier lorsqu'ils comportent des choix entre diverses évolutions possibles ou des séquences à déroulement simultané. C'est pourquoi il est utile de disposer d'un outil de représentation d'un cahier des charges qui soit normalisé, dépourvu d'ambiguïtés et cependant facile à comprendre et à utiliser.

Le GRAFCET, décrit au chapitre suivant, se propose de répondre à de telles exigences.

II

LE GRAFCET

Le GRAFCET est un outil de description du cahier des charges de la partie commande du système automatisé utilisable tant au niveau 1 qu'au niveau 2.

Le fonctionnement de l'automatisme peut être représenté graphiquement par un ensemble :

- **D'ÉTAPES** auxquelles sont associées des **ACTIONS**
- **DE TRANSITIONS** auxquelles sont associées des **RÉCEPTIVITÉS**
- **DE LIAISONS ORIENTÉES** reliant les étapes aux transitions et les transitions aux étapes

Avant d'introduire ces concepts et leur représentation nous allons montrer leur importance ainsi que leur signification pratique à partir d'un exemple simplifié.

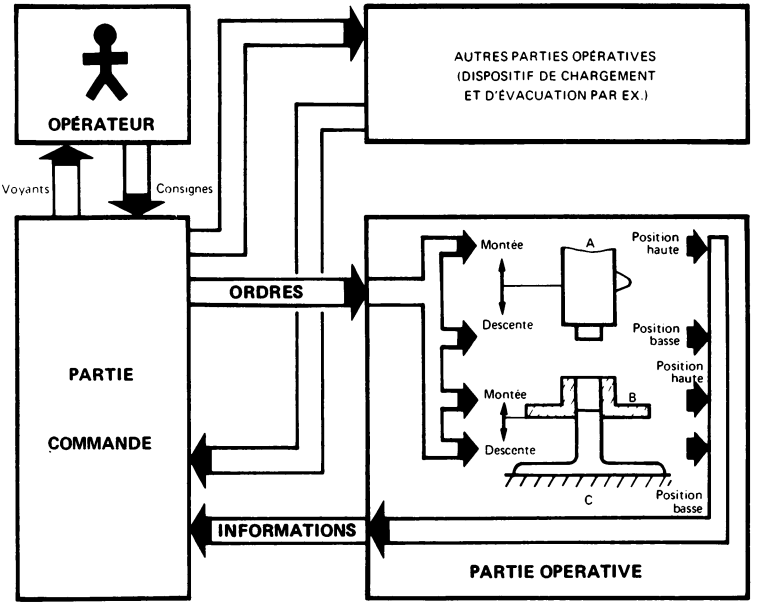
2.1 EXEMPLE INTRODUCTIF : PRESSE DE COMPRESSION DE POUDRES

On se propose d'étudier l'automatisation d'une presse destinée à la fabrication de pièces à partir de poudres comprimées.

2.1.1 Découpage Partie Opérative – Partie Commande

La partie opérative représentée ci-dessous très schématiquement se compose :

- d'un poinçon inférieur fixe C
- d'un poinçon supérieur A et d'une matrice B mobiles
- d'un sous-ensemble de mise en place de la matière
- d'un sous-ensemble d'évacuation de la pièce comprimée



2.1.2 Fonctionnement général du système

Le cycle de travail est le suivant :

- la matrice étant en haut de sa course, le poinçon inférieur qui y demeure engagé, délimite au-dessus de lui un espace suffisant pour recevoir la matière à comprimer. Le poinçon supérieur est alors dans sa position la plus haute ce qui dégage la partie supérieure de la matrice et permet l'introduction de la matière.
- quand la matière pulvérulente est en place, le poinçon supérieur descend, comprime la matière en pénétrant dans la matrice puis remonte en position haute.
- la matrice descend alors jusqu'à ce que le poinçon inférieur affleure, ce qui libère la pièce qui vient d'être comprimée. Cette pièce peut ensuite être évacuée.
- enfin la matrice reprend sa place et un nouveau cycle peut alors commencer.

Ces **actions** ne pourront être obtenues que si la partie commande émet les **ordres** convenables **au moment voulu.**

Les moments voulus seront déterminés d'après les compte rendus ou **informations** provenant de la partie opérative.

2.1.3 Etude de la partie commande

Considérons la presse arrêtée dans l'attente d'une nouvelle charge de matière. La matrice et le poinçon sont immobiles et la descente de ce dernier ne sera commandée par l'automate qu'après la réception de l'information "matière en place". Cependant, cette même information, si elle est renouvelée par erreur pendant la remontée du poinçon, n'aura aucun effet sur le comportement de la partie commande. Nous dirons que l'automate était "réceptif" dans le premier cas pour l'information "matière en place" et qu'il ne l'était plus dans le second.

Nous dirons que la partie commande demeure dans une "étape" tant que son comportement est constant. Elle reste dans cette "étape" jusqu'à ce que les informations pour lesquelles elle est "réceptive" provoquent le franchissement d'une "transition" conduisant à une nouvelle étape où la partie commande adoptera alors un nouveau comportement.

Nous pouvons maintenant représenter le fonctionnement d'une partie commande comme une **succession alternée** d'étapes et de transitions.

En conséquence nous associerons :

- à chaque **étape**, les **actions** à effectuer.
- à chaque **transition**, les informations permettant leur franchissement, sous forme d'une condition logique appelée **réceptivité.**

De cette manière le fonctionnement de la partie commande nécessaire à la presse sera décrit ainsi :

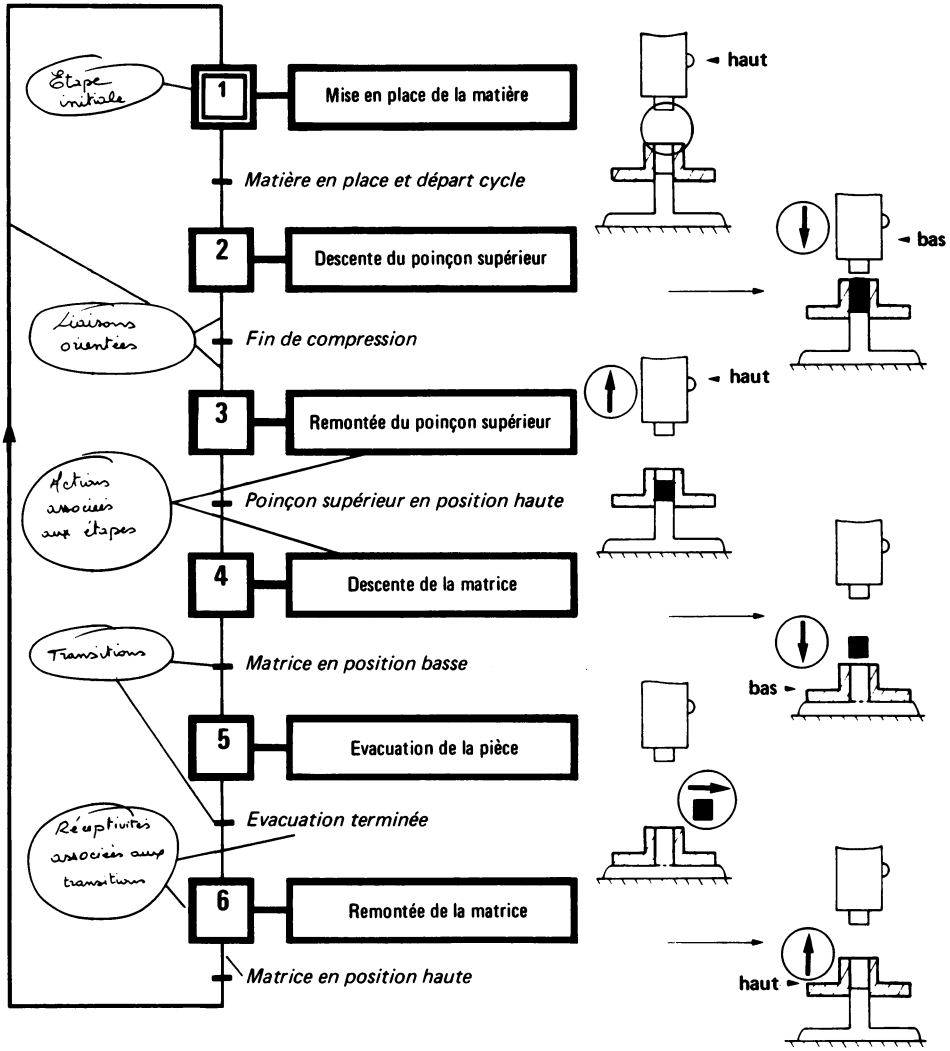
Etape 1 ; action : mise en place de la matière
Transition 1-2 ; réceptivité : matière en place et départ cycle
Etape 2 ; action : descente du poinçon
Transition 2-3 ; réceptivité : fin de compression
Etape 3 ; action : remontée du poinçon
Transition 3-4 ; réceptivité : poinçon en haut
Etape 4 ; action : descente matrice
Transition 4-5 ; réceptivité : matrice en bas
Etape 5 ; action : évacuation de la pièce comprimée
Transition 5-6 ; réceptivité : pièce évacuée
Etape 6 ; action : remontée matrice
Transition 6-1 ; réceptivité : matrice en haut

Il s'avère plus commode de représenter ce fonctionnement sous forme graphique d'où le GRAFCET ci-contre, illustré à des fins de compréhension des états correspondants de la partie opérative.

Aux actions ci-dessus, strictement nécessaires au fonctionnement de la presse, pourraient s'ajouter d'autres actions vers le monde extérieur telles l'allumage de voyants, etc. (par exemple "appel de l'opérateur pour évacuer la pièce" dans l'étape 5).

Enfin notons que nous avons attribué un rôle particulier à l'une des étapes : l'étape **initiale.** Le choix de cette étape est imposé par des considérations fonctionnelles liées à la partie opérative.

GRAFCET DE NIVEAU 1 DE LA PRESSE



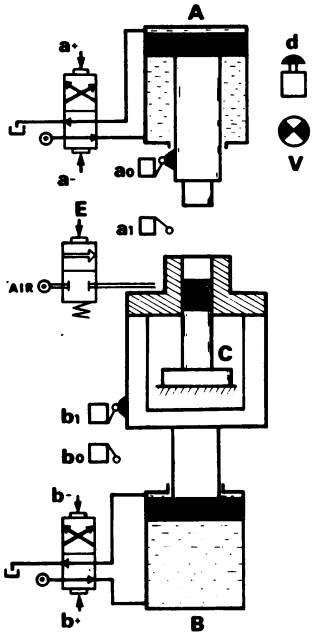
Le GRAFCET montre :

- les **liaisons** d'étape à transition et de transition à étape (Les liaisons non fléchées sont implicitement orientées du haut vers le bas)
- les **étapes** et leurs actions associées
- les **transitions** et leur réceptivité associée

2.1.4 Passage au niveau 2

Le GRAFCET que nous venons d'établir est un GRAFCET de niveau 1 (cf. chapitre 1) car il ne prend en compte que l'aspect fonctionnel sans aucune implication technologique : par exemple nous ne savons pas comment physiquement donner l'ordre de descente au poinçon, ni comment on s'assure que la pièce est évacuée.

Il convient maintenant de préciser les choix technologiques des actionneurs et des capteurs :



- cet exemple étant simplifié la procédure d'arrêt d'urgence, les modes de marche ainsi que les sécurités ne sont pas traités.

- la mise en place de la matière est assurée manuellement par l'opérateur. Un voyant V est allumé pendant toute la durée de la mise en place. Celle-ci terminée l'opérateur autorise la poursuite des opérations en appuyant sur un bouton-poussoir d.

- les mouvements du poinçon supérieur et de la matrice sont effectués au moyen de vérins hydrauliques double-effet. Les positions haute et basse du poinçon et de la matrice sont contrôlées à l'aide de capteurs de fin de course (respectivement : a_0 et a_1 , b_1 et b_0).

- l'évacuation de la pièce est obtenue au moyen d'un jet d'air maintenu pendant une seconde. Ce jet d'air est commandé par une électrovanne E.

La liste ci-après rappelle les variables introduites ainsi que leur signification respective.

Il est commode en pratique de les présenter sous forme d'un tableau des informations et des actions de la partie commande.

Nous pouvons alors établir pour la partie commande le GRAFCET ci-après de niveau 2.

GRAFCET DE NIVEAU 2 DE LA PRESSE

ORDRES

vers le milieu extérieur et l'opérateur

V : voyant "Prêt"

commande des actionneurs

a + : descente poinçon
a — : remontée poinçon
b — : descente matrice
b + : remontée matrice
E : évacuation

lancement de temporisations

LT1 : lancement temporisation d'évacuation

INFORMATIONS

déroulement du cycle

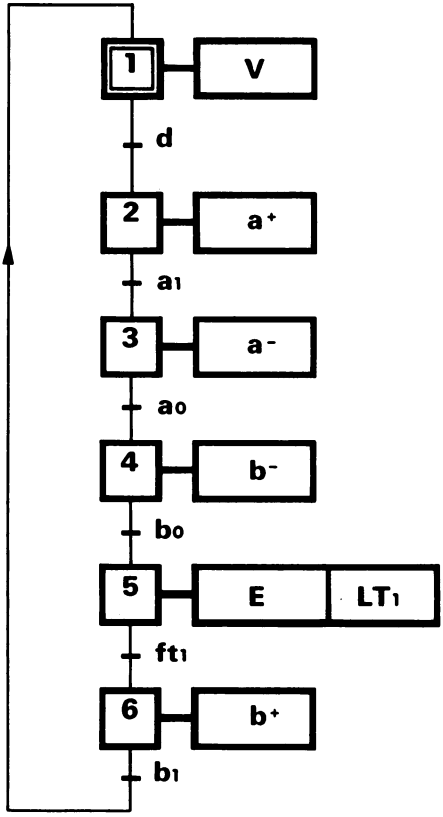
d : autorisation de départ cycle

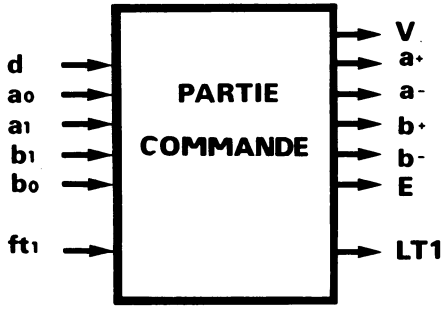
fin de course des actionneurs

a_1 : position basse du poinçon
a_0 : position haute du poinçon
b_0 : position basse de la matrice
b_1 : position haute de la matrice

fin de temporisation

ft_1 : fin de temporisation d'évacuation





Nota : Les arrêts d'urgence, les modes de marche et les sécurités ne sont pas traités.

2.2 ÉLÉMENTS DU GRAFCET

L'exemple simplifié précédent a permis de présenter de manière intuitive les trois concepts fondamentaux du GRAFCET : étape - transition - liaisons orientées. Nous allons maintenant en donner des définitions plus précises.

2.2.1 Étape

Une ÉTAPE correspond à une situation dans laquelle le comportement de tout ou partie du système par rapport à ses entrées et ses sorties est invariant.

L'ÉTAPE se représente par un carré ou un rectangle repéré numériquement, le repère étant placé à la partie supérieure.

En addition à ce repère, un nom symbolique peut être adjoint, représentatif de la fonction principale de l'étape (ex : ATTENTE, FIN, SYNCHRONISATION, etc.)

Une étape est soit active soit inactive et à un instant donné la situation du système automatisé est entièrement définie par l'ensemble des étapes actives. On précise pour chaque étape les actions à effectuer caractéristiques de cette situation. Ces actions ne sont effectives que lorsque l'étape est active.

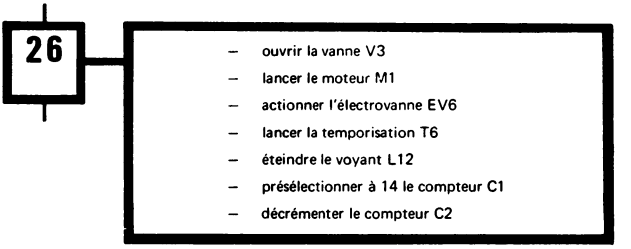
Ex :

Il est commode de montrer les étapes actives à un instant bien précis en plaçant un point ou un repère quelconque dans la partie inférieure des symboles correspondants.

Au niveau des spécifications fonctionnelles, on ne définit pas les actionneurs ni les capteurs, mais uniquement les actions à effectuer et leur enchaînement.

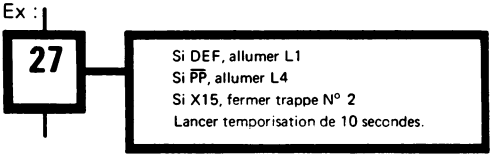
Les actions à effectuer lorsque l'étape est active sont décrites de façon littérale ou symbolique à l'intérieur d'un ou plusieurs rectangles de dimensions quelconques reliés à la partie droite de l'étape.

Ces actions peuvent être de nature fort diverses:



De plus l'exécution de ces actions peut être soumise à d'autres conditions logiques, fonction de variables d'entrée, de variables auxiliaires ou de l'état actif ou inactif d'autres étapes.

Ex :



Lorsque l'étape 27 est active, il faut :
- allumer L1 si DEF est présent
- allumer L4 si PP est absent
- fermer la trappe n° 2 si l'étape 15 est active (X15 = 1)
- lancer une temporisation de 10 secondes
- etc.

Nota : on notera X_i la variable booléenne correspondant au caractère actif de l'étape i.

Au niveau des spécifications technologiques (niveau 2), on devra préciser la façon dont les actions sont réalisées compte tenu du matériel défini pour les capteurs et les actionneurs.
La prise en compte de ces nouvelles spécifications peut amener à modifier le GRAFCET de niveau 1.

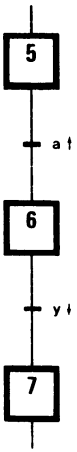
2.2.2 Transitions

Les **TRANSITIONS** indiquent les possibilités d'évolution entre étapes. On associe à chaque transition une condition logique appelée **RÉCEPTIVITÉ** qui permet de distinguer, parmi toutes les informations disponibles, uniquement celles qui sont susceptibles à un instant donné de faire évoluer la partie commande.

La **RÉCEPTIVITÉ**, écrite sous forme de proposition logique, est une fonction combinatoire d'informations extérieures (directives de l'opérateur, états de capteurs, de compteurs, de temporisateurs, etc.), de variables auxiliaires ou de l'état actif ou inactif d'autres étapes.

Ces réceptivités peuvent s'exprimer sous des formes diverses :

Ex : - position droite d'un mobile
- fin de course de trappe ouverte actionné
- température > 300° C
- valeur du compteur C 10 > valeur de consigne
- on a appuyé 3 fois sur le bouton M, etc.



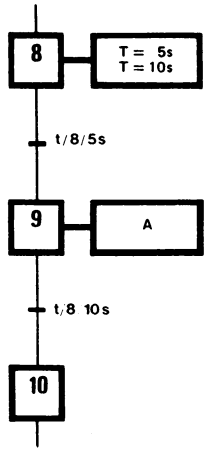
Les réceptivités peuvent aussi faire intervenir des changements d'état de variables.

La notation a↑ représente le front "montant" de la variable a (passage de l'état logique "0" à l'état logique "1") et la notation y↓ représente le front "descendant" de la variable y (passage de l'état logique "1" à l'état logique "0").

Pour faire intervenir le temps dans une réceptivité, il suffit d'indiquer après le repère t son origine et sa durée; l'origine sera l'instant du début de la dernière activation d'une étape antérieure.

Ex : la notation t/8/10 s signifie 10 secondes écoulées depuis la dernière activation de l'étape 8.

Lorsqu'une étape se trouve être à l'origine d'un temps, il peut être utile de l'indiquer comme une action associée à cette étape.



Nota : Une réceptivité toujours vraie est écrite = 1.

2.2.3 Liaisons orientées

Les liaisons indiquent les voies d'évolution de l'état du GRAFCET.

Les liaisons sont horizontales ou verticales sauf dans des cas isolés où des traits obliques apporteraient de la clarté au diagramme. L'essentiel est d'adopter une représentation qui contribue au mieux à la clarté du fonctionnement (Cf. norme NFZ 67-010).

Le sens général de parcours est du haut vers le bas. L'arrivée et le départ sur une étape sont représentés verticalement, l'arrivée étant à la partie supérieure. Si dans des cas très particuliers, l'arrivée devait être faite à la partie inférieure une flèche serait obligatoire.

Des flèches doivent être utilisées chaque fois qu'une meilleure compréhension pourra en résulter et chaque fois que l'orientation fixée n'est pas respectée.

Pour éviter toute ambiguité, il est préférable d'éviter les croisements continus des lignes de liaison.

2.3 REGLES D'ÉVOLUTION

Le caractère actif ou inactif de chacune des étapes devant évoluer, les trois concepts précédents ne peuvent suffire à définir un GRAFCET. Il est en plus nécessaire de fixer un ensemble de règles d'évolution.

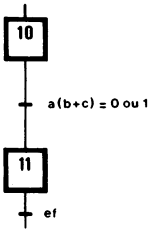
Règle 1 : L'**INITIALISATION** précise les étapes actives au début du fonctionnement. Elles sont activées inconditionnellement et repérées sur le GRAFCET en doublant les côtés des symboles correspondants.

Règle 2 : Une **TRANSITION** est soit validée soit non validée. Elle est validée lorsque **TOUTES** les étapes immédiatement précédentes sont actives. Elle ne peut être franchie que :

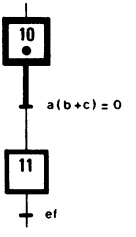
- lorsqu'elle est validée,
- ET que la réceptivité associée à la transition est vraie.

La **TRANSITION** est alors obligatoirement franchie.

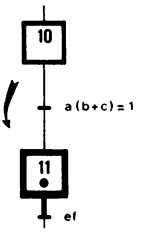
Ex :



Transition non validée
La transition 10-11 est non validée, l'étape 10 étant inactive.



Transition validée
La transition 10-11 est validée, l'étape 10 étant active, mais ne peut être franchie car la réceptivité a(b +c) = 0

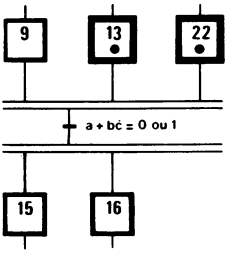


Transition franchie
La transition 10-11 est franchie car la réceptivité a(b +c) = 1

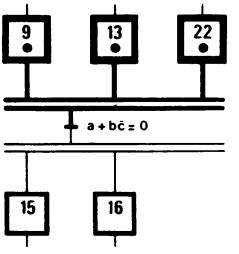
Règle 3 : Le franchissement d'une **TRANSITION** entraîne l'activation de TOUTES les étapes immédiatement suivantes et la désactivation de TOUTES les étapes immédiatement précédentes.

Exemple : Cas de transition entre plusieurs étapes

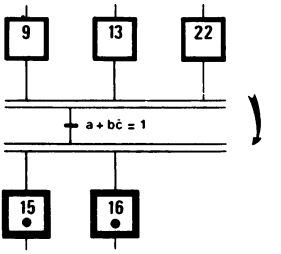
Lorsque plusieurs étapes sont reliées à une même transition on convient, pour des raisons pratiques, de représenter le regroupement de liaisons par deux traits parallèles (cf. normes NFZ 67.010 - ISO 1028).



Transition non validée
(étape 9 inactive)



Transition validée
(9, 13, 22 actives)



Transition franchie
(9, 13, 22 inactives; 15, 16 actives)

Règle 4 : Plusieurs transitions simultanément franchissables sont simultanément franchies.

Règle 5 : Si au cours du fonctionnement une même étape doit être désactivée et activée simultanément, elle reste activée.

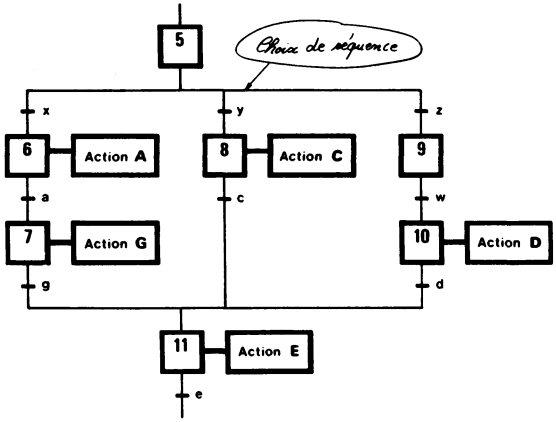
Nota : La durée de franchissement d'une transition ne peut jamais être rigoureusement nulle, même si, théoriquement (règles 3 et 4), elle peut être rendue aussi petite que l'on veut. Il en est de même de la durée d'activation d'une étape. En outre, la règle 5 se rencontre très rarement dans la pratique. Ces règles ont été ainsi formulées pour des raisons de cohérence théorique interne au GRAFCET.

2.4 REPRÉSENTATION DES SÉQUENCES MULTIPLES

2.4.1 Les aiguillages :
Choix conditionnel entre plusieurs séquences

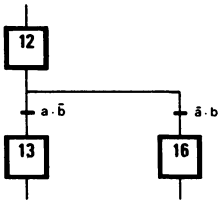
Un GRAFCET est généralement constitué de plusieurs séquences, c'est-à-dire de plusieurs suites d'étapes à exécuter les unes après les autres et il est souvent nécessaire d'effectuer une sélection exclusive d'une parmi ces séquences.

Ex :

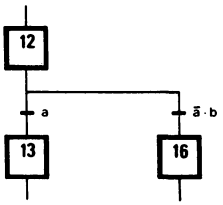


Dans l'aiguillage formé par le choix de la séquence à réaliser, les différentes transitions correspondant aux réceptivités x, y et z étant simultanément validées par la même étape 5 pourraient, d'après la règle 4 de simultanéité, être franchies simultanément. En pratique, on est souvent amené à rendre ces réceptivités exclusives. On peut également introduire des priorités.

Ex :



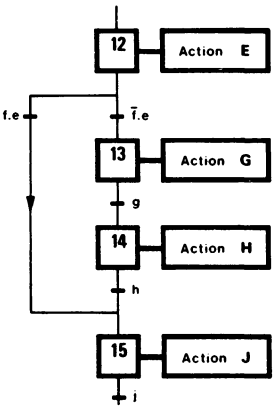
Réceptivités $a.\bar{b}$ et $\bar{a}.b$ exclusives
Si a et b sont présents à la fois, aucune transition ne pourra être franchie à partir de l'étape 12.



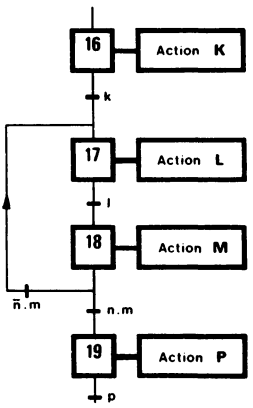
Priorité à la réceptivité a. La priorité donnée à la transition 12-13 permet à celle-ci d'être franchie lorsque a et b sont présents en même temps.

Saut d'étapes et reprise de séquence

Le saut conditionnel est un aiguillage particulier permettant de sauter une ou plusieurs étapes lorsque les actions à réaliser deviennent inutiles, tandis que la reprise de séquence permet au contraire de reprendre une ou plusieurs fois la même séquence tant qu'une condition fixée n'est pas obtenue.



Saut de l'étape 12 à l'étape 15 par la réceptivité f.e

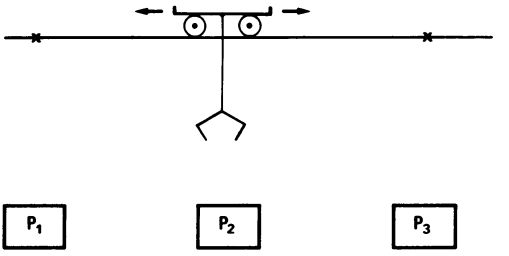


Reprise de la séquence 17-18 par la réceptivité $\bar{n}.m$ tant que la réceptivité n.m ne s'effectue pas.

Premier exemple d'aiguillage : desserte de 3 postes

Soit un dispositif de manipulation pouvant desservir 3 postes P1, P2 et P3.

Au repos, le dispositif est présent à l'un des 3 postes pince ouverte.



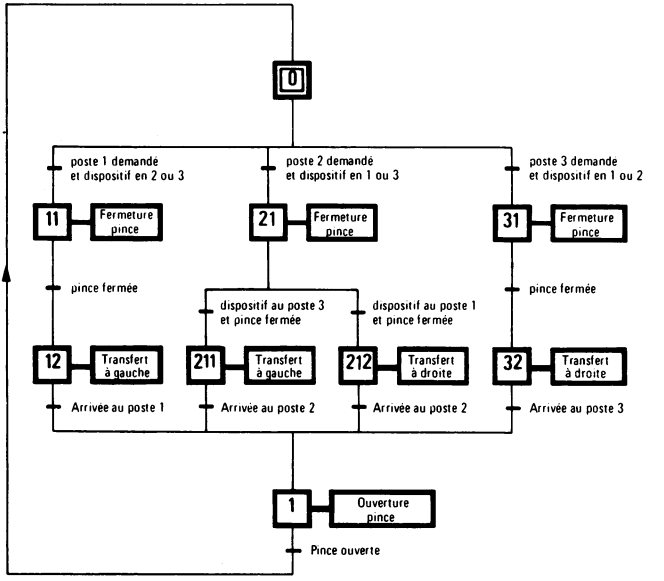
Un pupitre comprend 3 boutons-poussoir correspondant à des demandes de transfert vers l'un des trois postes.

Lorsque le dispositif est au repos, la demande d'un autre poste déclenche la séquence suivante :

- fermeture de la pince (prise de l'objet)
- transfert à gauche ou à droite suivant demande.
- ouverture de la pince dès que le poste désiré est atteint.

Pour des raisons de simplification on suppose qu'un seul appel peut être effectué à la fois.

GRAFCET DE NIVEAU 1

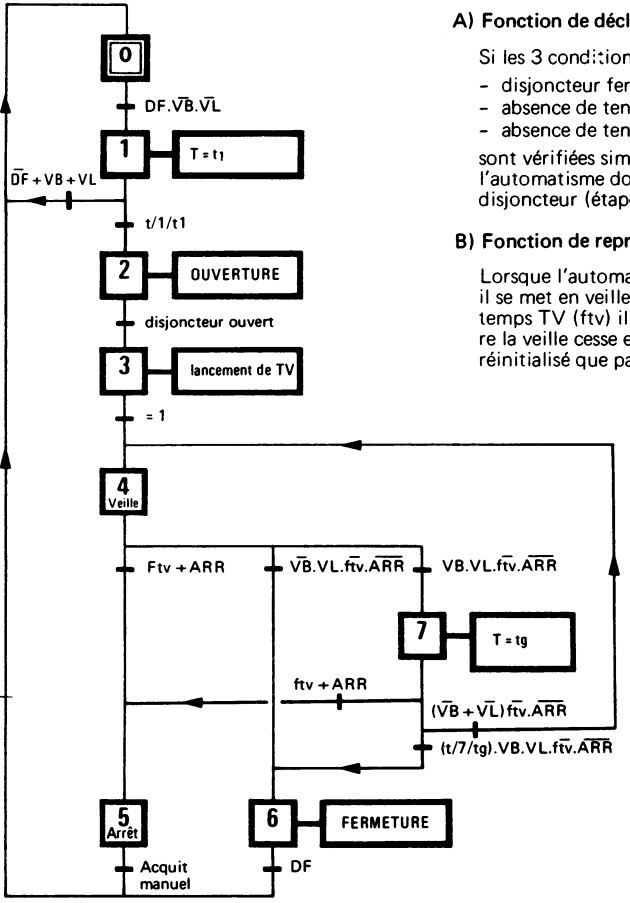


L'élaboration du GRAFCET de niveau 2 supposera résolus notamment les problèmes suivants :

- assurance de l'exclusivité des branches.
- comment se commande le transfert, etc.

Deuxième exemple d'aiguillage : automatisme à manque de tension

L'absence de tension de part et d'autre d'un disjoncteur fermé est une situation anormale et dangereuse. Le rôle d'un automatisme à manque de tension est de faire ouvrir le disjoncteur lorsqu'un tel cas se produit et ensuite de la refermer, moyennant certains contrôles, lorsque la tension revient d'un côté ou d'un autre.



A) Fonction de déclenchement

Si les 3 conditions suivantes :

- disjoncteur fermé (DF)
- absence de tension côté barres ($\overline{VB}$)
- absence de tension côté ligne ($\overline{VL}$)

sont vérifiées simultanément pendant un temps t_1, l'automatisme donne un ordre d'OUVERTURE au disjoncteur (étape 2).

B) Fonction de reprise

Lorsque l'automatisme a fait ouvrir le disjoncteur il se met en veille (étapes 3 et 4). Si au bout du temps TV (ftv) il n'a pu donner d'ordre de fermeture la veille cesse et le GRAFCET ne peut alors être réinitialisé que par une action manuelle (étape 5).

Une action extérieure (ARR) produit le même effet. Par contre, si l'une des tensions revient pendant cette veille (VL ou VB) deux séquences sont envisagées.

Renvoi

Si VL revient en premier on referme le disjoncteur et on réinitialise le GRAFCET (étape 6).

Rebouclage

Le retour de VB et VL lancent une seconde temporisation TG (étape 7).

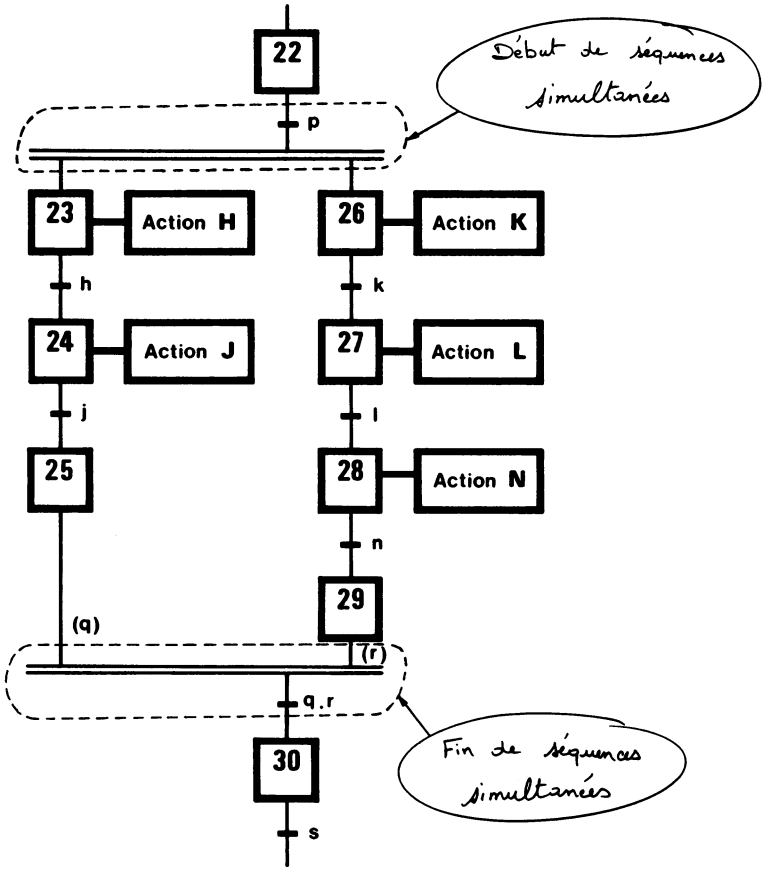
Si les deux tensions restent présentes pendant le temps tg on referme le disjoncteur (étape 6).

Sinon l'automatisme se remet en veille (étape 4). Enfin, l'échéance de TV (ftv) provoque l'arrêt de l'automatisme (étape 5) dans l'attente d'une réinitialisation manuelle.

2.4.2 Séquences simultanées

Un GRAFCET peut comporter plusieurs séquences s'exécutant simultanément mais dont les évolutions des étapes actives dans chaque branche restent indépendantes.

Pour représenter ces fonctionnements simultanés, une transition UNIQUE et deux traits parallèles indiquent le début et la fin des séquences, c'est-à-dire l'activation simultanée des branches ainsi réalisées et leur attente réciproque vers une séquence commune.



A partir de l'étape 22, la réceptivité p provoque l'activation simultanée des étapes 23 et 26. Ces deux séquences 23-24-25 et 26-27-28-29 évolueront alors de façon totalement indépendante et ce n'est que :

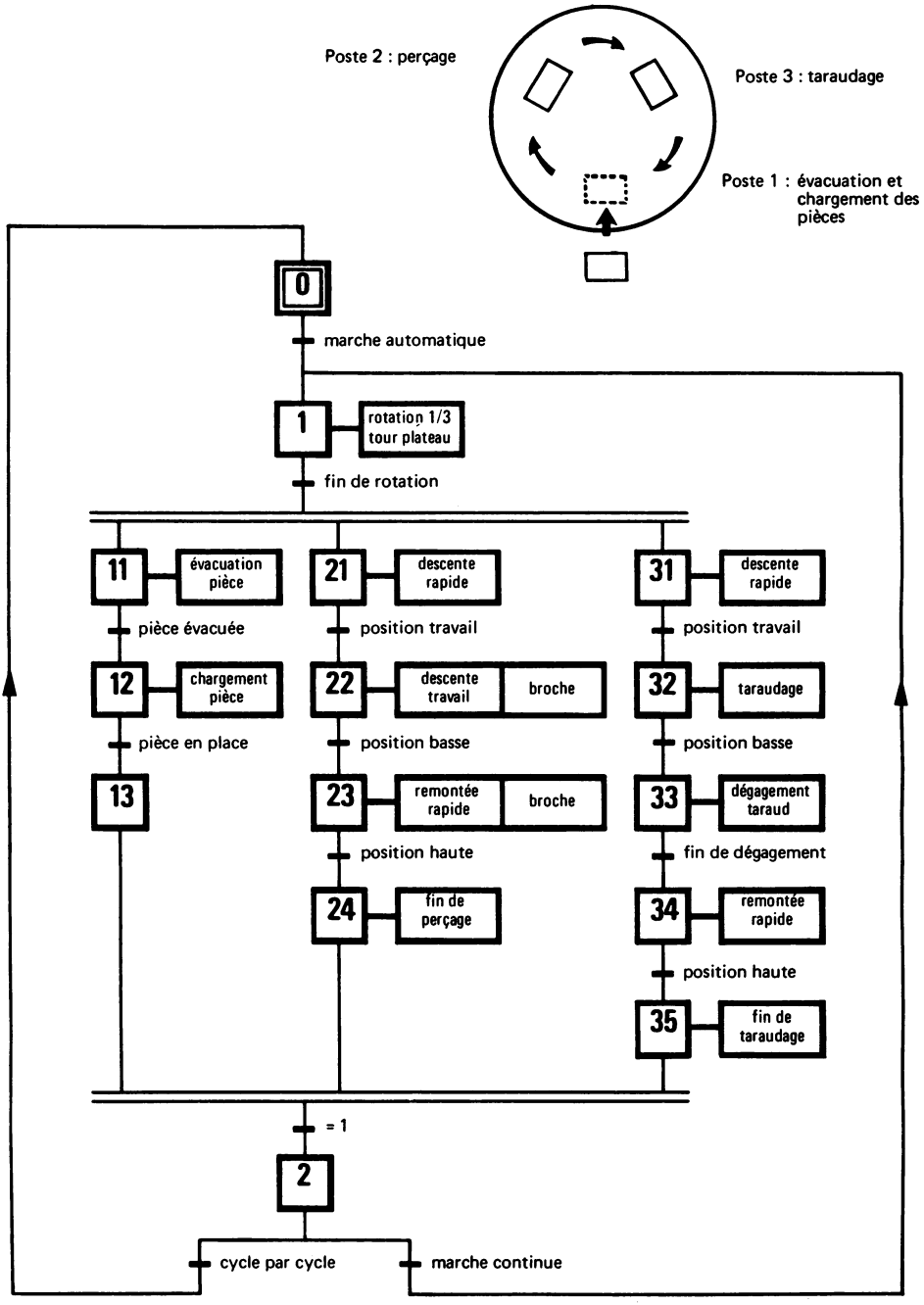
- lorsque les étapes de fin de branche 25 et 29 étant actives,
- lorsque la réceptivité étant vraie (q.r = 1),

que la transition sera franchie.

L'étape 30 devient alors active et les étapes 25 et 29 inactives.

Nota : les conditions particulières à chaque branche peuvent être notées entre parenthèses au-dessus des traits parallèles de regroupement.

Exemple de séquences simultanées : GRAFCET de niveau 1 d'une unité de perçage — taraudage



BIBLIOGRAPHIE

(1) **AFCET**
"Rapport de la Commission de Normalisation de la Représentation du Cahier des Charges d'un Automatisme Logique"
Automatisme Tome XXIII n° 3 - 4 Mars Avril 1978 pp 66 - 83
Automatique et Informatique Industrielle n° 61 et 62 Novembre Décembre 1977

(2) **M. BLANCHARD**
"Approche "unitaire" de la conception des systèmes logiques"
L'Ingénieur et le Technicien de l'Enseignement Technique n° 215 Novembre et Décembre 1978

(3) **M. BLANCHARD**
"GRAFCET ou réseau de Petri ? "
Le Nouvel Automatisme (à paraître)

(4) **D. BOUTEILLE**
"Un diagramme fonctionnel au service des automatismes pneumatiques"
Energie fluide n° 104 Juin 1978

(5) **R. CASTELLANET**
"Informatique et Automatique"
L'Ingénieur et le Technicien de l'Enseignement Technique n° 216 Janvier - Février 1979

(6) **R. DAVID**
"Synthèse à l'aide de CUSA d'un système séquentiel décrit par un GRAFCET"
Journée SEE 2 - 3 Février 1978 - GIF SUR YVETTE

(7) **B. TACONET – B. CHOLLOT**
"Programmation du GRAFCET sur automate programmable à langage logique, à relais ou booleen"
Le Nouvel Automatisme n° 4 Janvier - Février 1979

(8) **L. TOURRES**
"Le GRAFCET, outil de représentation du Cahier des Charges d'un automatisme logique"
L'Ingénieur et le Technicien de l'Enseignement Technique n° 213 - Juillet - Août 1978

(9) **L. TOURRES**
"Le GRAFCET"
L'Ingénieur et le Technicien de l'Enseignement Technique n° 214 Septembre - Octobre 1978

(10) **R. VALETTE**
"Etude comparative de deux outils de représentation : GRAFCET et Réseau de Petri"
Le Nouvel Automatisme n° 3 Décembre 1978

Appendice B

Corrigé des problèmes de numéros impairs

CHAPITRE 1

1-1 a) $27 = 2 \times 10^1 + 7 \times 10^0$

b) $301 = 3 \times 10^2 + 0 \times 10^1 + 1 \times 10^0$

c) $08641 = 0 \times 10^4 + 8 \times 10^3 + 6 \times 10^2 + 4 \times 10^1 + 1 \times 10^0$

d) $39472 = 3 \times 10^4 + 9 \times 10^3 + 4 \times 10^2 + 7 \times 10^1 + 2 \times 10^0$

e) $473283 = 4 \times 10^5 + 7 \times 10^4 + 3 \times 10^3 + 2 \times 10^2 + 8 \times 10^1 + 3 \times 10^0$

1-3 a) 2; b) 0; c) 3

1-5 a) $\{0, 1, 2\}$;

b) $\{0, 1, 2, 3, 4, 5, 6\}$;

c) $\{0, 1, 2, 3,\}$;

d) $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A\}$;

e) $\{0, 1, 2, 3, 4\}$;

f) $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C\}$;

g) $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$

1-7 a) 3281: b) 373; c) 3545; d) 4112; e) 33

1-9 a) 135476_8, $BB3E_{16}$;

b) 261774_8, 163 FC_{16};

c) 1347_8, $2E7_{16}$;

d) 171_8, 79_{16}

1-11 a) 75310_8; $7AC8_{16}$;

b) 253613_8; $1578B_{16}$;

c) 1215_8: $28D_{16}$;

d) 1477_8; $33F_{16}$.

1-13 a) 3,421875; b) 1,703125; c) 1,0859375

1-15 a) $0{,}011_2$; $0{,}3_8$; $0{,}6_{16}$;

b) $0{,}011011101_2$; $0{,}335_8$; $0{,}6E9_{16}$;

c) $0{,}10111101_2$; $0{,}572_8$; $0{,}BD_{16}$.

1-17 a) $28D_{16}$: $10\ 1000\ 1101_2$;

b) $33F_{16}$; $11\ 0011\ 1111_2$;

c) $1578B_{16}$; $1\ 0101\ 0111\ 1000\ 1011_2$.

1-19 a) 10010,1000;

b) 1101011,00;

c) 1101111000;

d) 11011111.

1-21 a) 111011,0001;

b) 10001110,1;

c) 101100011,01;

d) 1110100111.

1-23 A) a) 0010001000;

b) 0100100010;

c) 00100;

d) 01001000;

e) 0111110;

f) 0011000;

g) 001000;

h) 01001010;

i) 10110001.

B) a) 0010001001;

b) 0100100011;

c) 00101;

d) 01001001;

e) 0111111;

f) 0011001;

g) 001001;

h) 01001011;

i) 10110010.

1-25 a) 1110000 + 0010001 = 1) 0000001 (à éliminer)

b) 1101111 + 0100011 = 1) 0010010 (à éliminer)

c) 100111101 + 100100010 = 1) 001011111 (à éliminer)

1-27 0 pour +; 1 pour −.

1-29 a) 10110; b) 10000; c) 11010; d) 11100.

1-31 a) 1000 0111 0011 0010;

b) 0100 0001 0100 1001

c) 0111 0000 0011 0010;

d) 0100 0000 1001 0110.

1-33 46_{16}; 100 0110.

1-35 80

1-37 a) 2 fils: (un de masse);

b) 6 fils;

c) le mode parallèle.

CHAPITRE 2

2-1 a) $S = A + (B + C) = (A + B) + C$

2-3 a) $S = A + BC = (A + B)(A + C)$

2-5 $E = (a + d)(ab + ac)(\overline{\overline{a}}\,\overline{\overline{c}} + b)$

$= (aab + aac + dab + dac)(\overline{\overline{a}}\,\overline{\overline{c}} + b)$

$= (ab + ac + dab + dac)(\overline{\overline{a}}\,\overline{\overline{c}} + b)$

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

$= a b \bar{a}\bar{c} + ab + ac\bar{a}\bar{c} + acb$

$= ab + acb$

$= ab$

2-7 $E = (a + c + d)(b + c + d)$

$= ab + ac + ad + cb + cc + cd + db + dc + dd$

$= ab + ac + ad + cb + c + cd + db + dc + d$

$= ab + c + d$

2-9 $E = (a\bar{b} + ab + ac)(\bar{a}\bar{b} + ab + a\bar{c})$

$= a\bar{b}\bar{a}\bar{b} + a\bar{b}ab + a\bar{b}ac + ab\bar{a}\bar{b} + abab +$
$aba\bar{c} + ac\bar{a}\bar{b} + acab + aca\bar{c}$

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

$= ab + a\bar{c}$

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

2-11 $E = a(b\bar{c} + \bar{b}c)$

$= a(b \oplus c)$

2-13 $E = ab(a\bar{b}c + a\overline{bc} + ab\bar{c})$

$= aba\bar{b}c + aba(\bar{b} + \bar{c}) + ababc$

$= ab\bar{c} + ab\bar{c}$

$= ab\bar{c}$

2-15 $E = (a + b)(a + c)(\bar{a} + \bar{b})$

$= (aa + ac + ba + bc)(\bar{a} + \bar{b})$

$= (a + ac + ba + bc)(\bar{a} + \bar{b})$

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

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

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

2-17 $E = \bar{a}bc + a\bar{b}c + ab\bar{c}$ non simplifiable

2-19 $E = abc + \bar{a}bc + a\bar{b}c + ab\bar{c} + a\overline{bc} + \bar{a}b\bar{c} + \bar{a}\bar{b}c$

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

$= bc + a + a\bar{b}c + a\bar{b}c + \bar{a}b\bar{c}$

$= bc + a(1 + \bar{b}c) + (a + \bar{a})b\bar{c}$

$= bc + a + b\bar{c}$

$= a + b$

2-21 $E = (a + b + c)(a + \bar{b} + \bar{c})(a + b + \bar{c})(a + \bar{b} + c)$

$= (a + a\bar{b} + a\bar{c} + ba + b\bar{b} + b\bar{c} + ca + c\bar{b} + c\bar{c})$

$(a + a\bar{b} + ac + ba + b\bar{b} + bc + \bar{c}a + \bar{c}\bar{b} + \bar{c}c)$

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

$= a + b\bar{c}bc + b\bar{c}\bar{c}\bar{b} + c\bar{b}bc + c\bar{b}\bar{c}\bar{b}$

$= a$

2-23 $\bar{E} = \overline{(a + b)(b + c)(a + c)}$

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

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

2-25 $\bar{E} = \overline{a(b + c)(\bar{c} + \bar{d})}$

$= \bar{a} + \overline{b + c} + \overline{\bar{c} + \bar{d}}$

$= \bar{a} + \bar{b}\bar{c} + cd$

2-27 $D = a \cdot (b + c) \cdot (a + b) = a \cdot (b + c)$

2-29 $D = (a + b) + (\bar{c} + d)(\bar{b} + c) = a + b + \bar{c} + d$

CHAPITRE 3

3-1 $E = ab + a\bar{c}$

a) table de vérité dans l'ordre binaire naturel

c	b	a	$ab + a\bar{c}$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

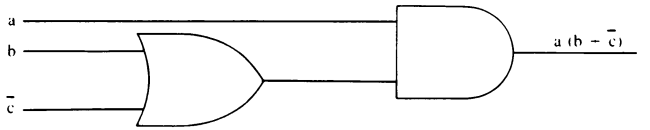
b) table de Karnaugh

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

E

forme simplifiée: $E = a\,(b + \bar{c})$

c) logigramme



Pour les problèmes 3-2, 3-3 et 3-4 suivants, seuls la table de Karnaugh, la forme simplifiée (si possible) et le logigramme sont donnés.

3-3 $E = a\,\overline{b\,c} + \bar{a}\,\overline{b\,c} + \bar{a}\,b\,\bar{c} + \overline{a\,b}\,c$

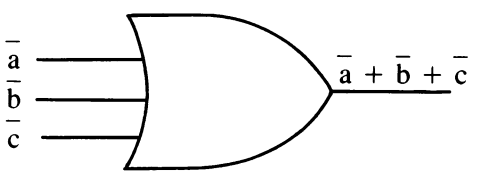
b) table de Karnaugh

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

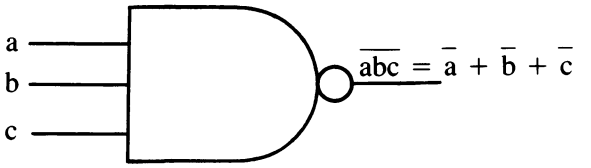
E

forme simplifiée: $E = \bar{a} + \bar{b} + \bar{c}$

c) logigramme



ou, selon le théorème de De Morgan



3-5 $E = (a + b)(\bar{a} + \bar{b} + \bar{c})(a + c)$

On a une expression sous la forme produit de sommes. Cette forme va nous permettre de dresser la table de Karnaugh de manière aisée. En effet si, par exemple, a= c = 0, alors E = 0. On obtient donc:

a) table de Karnaugh:

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

E

b) 1re forme canonique

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

c) 2e forme canonique

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

d) forme simplifiée

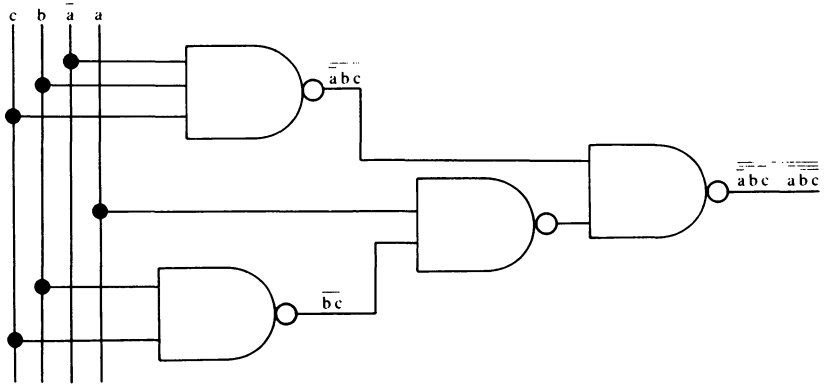
$E = a \oplus bc$

e) logigramme NON-ET

$E = \bar{a}bc + a\overline{bc}$

$= \overline{\bar{a}bc + a\overline{bc}}$

$= \overline{\bar{a}bc} \quad \overline{a\overline{bc}}$

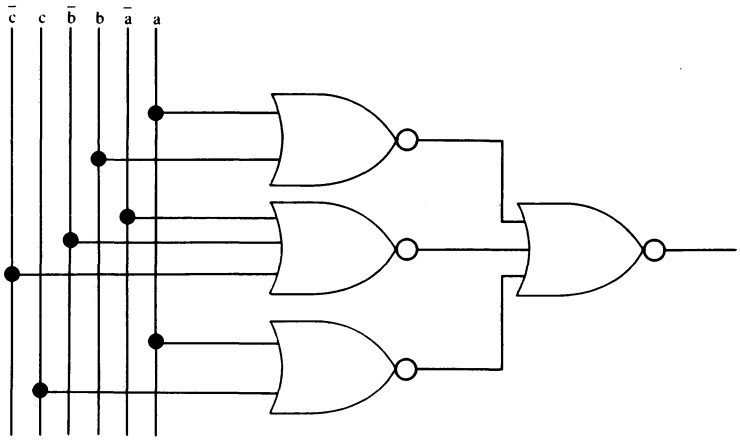


f) logigramme NON-OU

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

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

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



3-7 $E = (a + c + d)(b + c + d)$

a) table de Karnaugh

dc \ ba	00	01	11	10
00	0	0	1	0
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

E

b) 1ere forme canonique

$$E = \bar{a}\,\bar{b}\,c\,\bar{d} + \bar{a}\,\bar{b}\,c\,d + \bar{a}\,\bar{b}\,\bar{c}\,d + a\,\bar{b}\,c\,\bar{d} + a\,\bar{b}\,c\,d + a\,\bar{b}\,\bar{c}\,d + a\,b\,\bar{c}\,\bar{d} + a\,b\,c\,\bar{d} + a\,b\,c\,d + a\,b\,\bar{c}\,d + \bar{a}\,b\,c\,\bar{d} + \bar{a}\,b\,c\,d + \bar{a}\,b\,\bar{c}\,d$$

c) 2e forme canonique

$$E = (a + b + c + d)\,(\bar{a} + b + c + d)\,(a + \bar{b} + c + d)$$

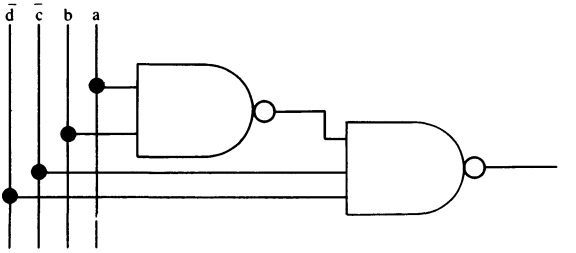
d) forme simplifiée

$$E = a\,b + c + d$$

e) logigramme NON-ET

$$E = \overline{\overline{a\,b + c + d}}$$

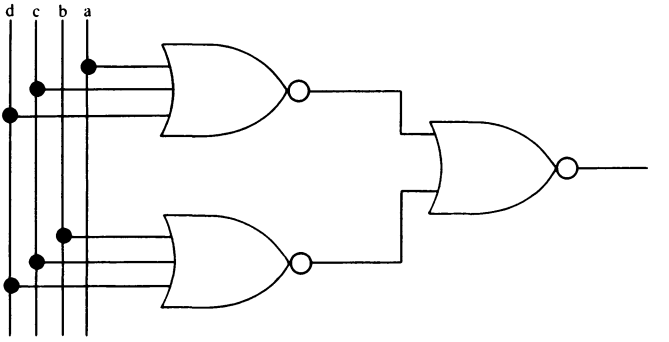
$$= \overline{\overline{a\,b}\quad\bar{c}\quad\bar{d}}$$



f) logigramme NON-OU

$$E = \overline{\overline{(a + c + d)\,(b + c + d)}}$$

$$= \overline{\overline{a + c + d} + \overline{b + c + d}}$$



3-9 $E = (a\bar{b} + ab + ac)(\bar{a}\bar{b} + ab + \bar{c})$

On aura après développement et simplification

$$E = ab + a\bar{c}$$

a) table de Karnaugh

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

b) 1re forme canonique

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

c) 2e forme canonique

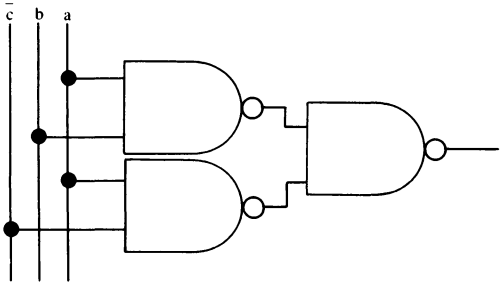
$$E = (a + b + c)(a + \bar{b} + c)(a + \bar{b} + \bar{c})(a + b + \bar{c})(\bar{a} + b + \bar{c})$$

d) forme simplifiée

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

e) logigramme NON-ET

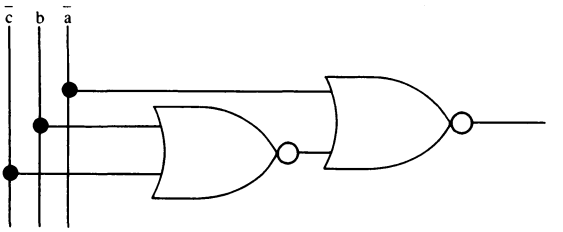
$$E = ab + a\bar{c}$$
$$= \overline{\overline{ab}\,\overline{a\bar{c}}}$$



f) logigramme NON-OU

$$E = a\,(b + \bar{c})$$

$$= \overline{\bar{a} + \overline{b + \bar{c}}}$$



3-11 $E = \bar{a}\,b\,c + a\,\bar{b}\,c + a\,b\,\bar{c}$

a) table de Karnaugh

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

E

b) 1ere forme canonique

$$E = \bar{a}\,b\,c + a\,\bar{b}\,c + a\,b\,\bar{c}$$

c) 2e forme canonique

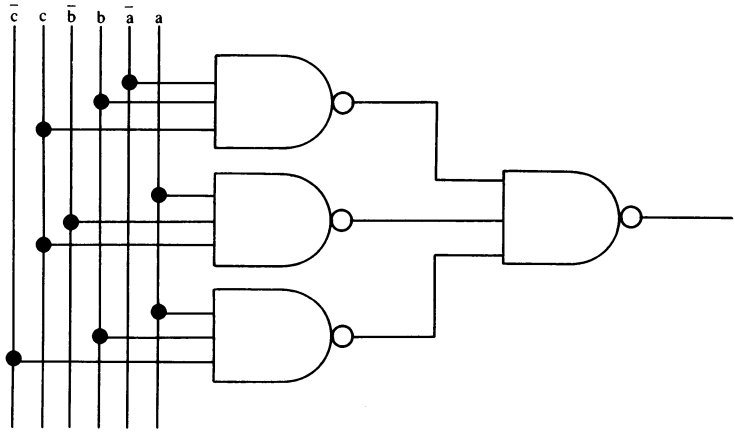
$$E = (a + b + c)\,(a + \bar{b} + c)\,(a + b + \bar{c})\,(a + b + c)\,(\bar{a} + \bar{b} + \bar{c})$$

d) Pas de simplification car il n'y a pas de cases 1 adjacentes.

e) logigramme NON-ET

$$E = \bar{a}\,b\,c + a\,\bar{b}\,c + a\,b\,\bar{c}$$

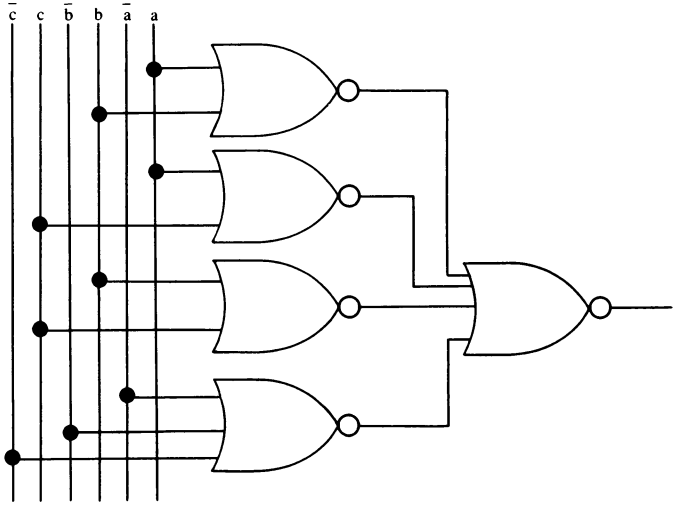
$$= \overline{\bar{a}\,b\,c} \quad \overline{a\,\bar{b}\,c} \quad \overline{a\,b\,\bar{c}}$$



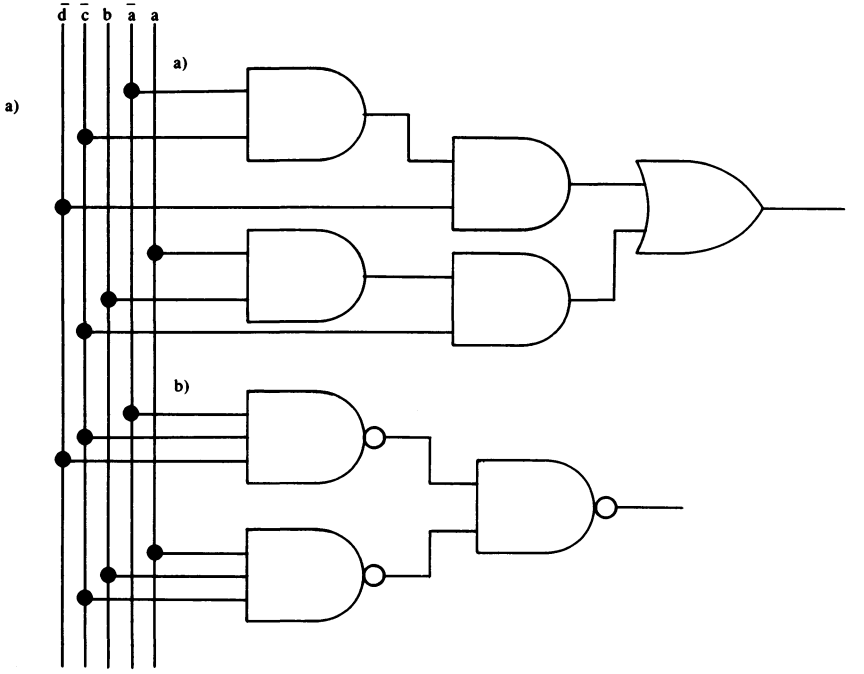
f) logigramme NON-OU

$E = (b + c)(a + c)(a + b)(\bar{a} + \bar{b} + \bar{c})$ selon la table de Karnaugh

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



3-13



3-15

dc \ ba	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	0
10	1	0	0	1

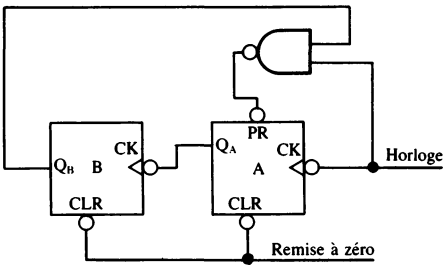
y

$$\bar{y} = a\bar{c} + \bar{a}bc + \bar{d}$$

3-17 $A\bar{B}\bar{C} + DE + \bar{A}E \equiv (\bar{A} + B + D)\,(\bar{A} + \bar{C} + D)\,(\bar{A} + B + E)$
$(\bar{A} + \bar{C} + E)\,(A + D + \bar{E})\,(B + D + \bar{E})\,(\bar{C} + D + \bar{E})$

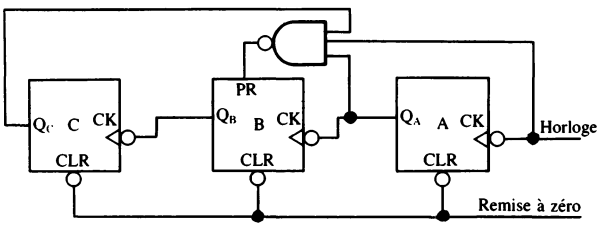
CHAPITRE 5

5-1



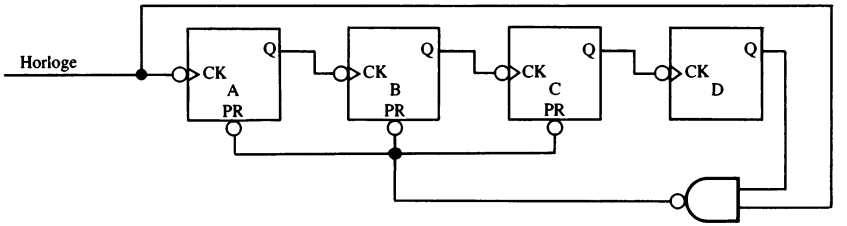
Q_B	Q_A
0	0
0	1
1	0
1	1

5-3



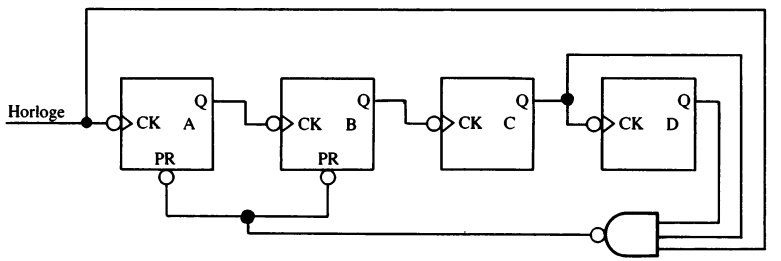
Q_C	Q_B	Q_A
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
0	1/0	0

5-5



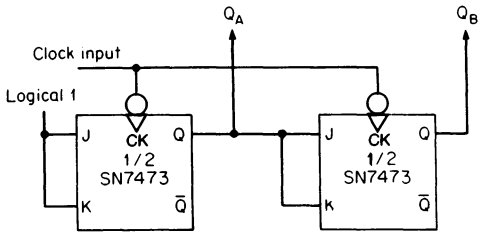
Q_D	Q_C	Q_B	Q_A
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
0	1/0	1/0	1/0

5-7



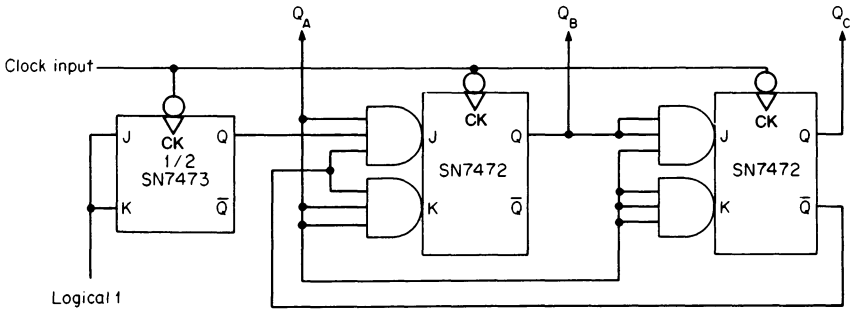
Q_D	Q_C	Q_B	Q_A
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
0	0	1/0	1/0

5-9



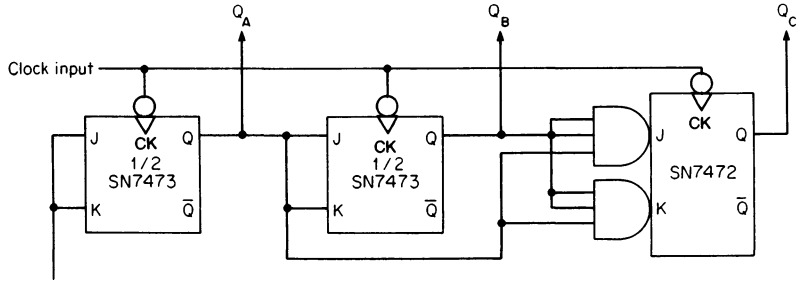
État	Q_A	Q_B
0	0	0
1	1	0
2	0	1
3	1	1

5-11



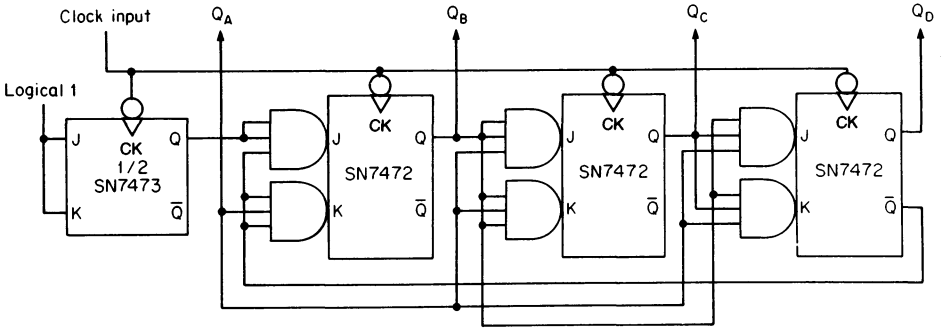
État	Q_A	Q_B	Q_C
0	0	0	0
1	1	0	0
2	0	1	0
3	1	1	0
4	0	0	1
5	1	0	1

5-13



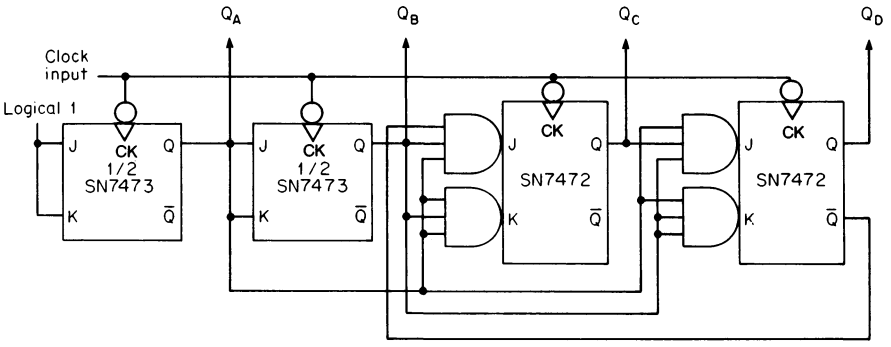
État	Q_A	Q_B	Q_C
0	0	0	0
1	1	0	0
2	0	1	0
3	1	1	0
4	0	0	1
5	1	0	1
6	0	1	1
7	1	1	1

5-15



État	Q_A	Q_B	Q_C	Q_D
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	1	1	0	0
4	0	0	1	0
5	1	0	1	0
6	0	1	1	0
7	1	1	1	0
8	0	0	0	1
9	1	0	0	1

5-17



État	Q_A	Q_B	Q_C	Q_D
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	1	1	0	0
4	0	0	1	0
5	1	0	1	0
6	0	1	1	0
7	1	1	1	0
8	0	0	0	1
9	1	0	0	1
10	0	1	0	1
11	1	1	0	1

Rappel de la table d'excitation des bascules JK pour les problèmes 5-19 et 5-21.

J	K	Q_n	Q_{n+1}
0	X	0	0
1	X	0	1
X	1	1	0
X	0	1	1

5-19

Q_DQ_C \ Q_BQ_A	00	01	11	10
00	X	X	0	X
01	1	2	4	3
11	9	X	X	X
10	5	6	8	7

J_A, K_A

Q_DQ_C \ Q_BQ_A	00	01	11	10
00	XX	XX	X1	XX
01	1X	X1	X1	1X
11	1X	XX	XX	XX
10	1X	X1	X1	1X

J_B, K_B

Q_DQ_C \ Q_BQ_A	00	01	11	10
00	XX	XX	X0	X0
01	0X	1X	X1	X0
11	1X	XX	XX	XX
10	0X	1X	1X	X0

$$J_A = K_A = 1$$

$$J_B = Q_DQ_C + Q_A = \overline{\overline{Q_DQ_C} \cdot \overline{Q_A}}$$

$$K_B = Q_A$$

J_C, K_C

Q_DQ_C \ Q_BQ_A	00	01	11	10
00	XX	XX	1X	XX
01	X0	X0	X1	X0
11	X1	XX	XX	XX
10	0X	0X	1X	0X

J_D, K_D

Q_DQ_C \ Q_BQ_A	00	01	11	10
00	XX	XX	0X	XX
01	0X	0X	1X	0X
11	X1	XX	XX	XX
10	X0	X0	X0	X0

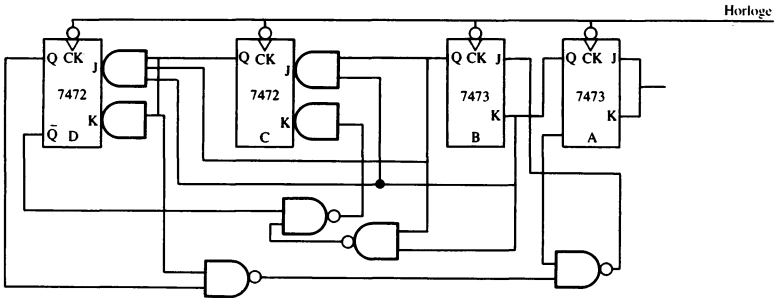
$$J_C = Q_AQ_B$$

$$K_C = Q_D + Q_AQ_B$$

$$= \overline{\overline{Q_D} \cdot \overline{Q_AQ_B}}$$

$$J_D = Q_AQ_BQ_C$$

$$K_D = Q_C$$



5-21

Q_DQ_C \ Q_BQ_A	00	01	11	10
00	0	1	3	2
01	4	X	X	X
11	6	7	9	8
10	X	X	5	X

J_A, K_A

Q_DQ_C \ Q_BQ_A	00	01	11	10
00	1X	X1	X1	1X
01	1X	XX	XX	XX
11	1X	X1	X1	1X
10	XX	XX	X1	XX

J_B, K_B

Q_DQ_C \ Q_BQ_A	00	01	11	10
00	0X	1X	X1	X0
01	1X	XX	XX	XX
11	0X	1X	X1	X0
10	XX	XX	X1	XX

$$J_A = K_A = 1$$

$$J_B = Q_A + Q_C\overline{Q_D} = \overline{\overline{Q_A} \cdot \overline{Q_C\overline{Q_D}}}$$

$$K_B = Q_A$$

J_C, K_C

Q_DQ_C \ Q_BQ_A	00	01	11	10
00	0X	0X	1X	0X
01	X1	XX	XX	XX
11	X0	X0	X1	X0
10	XX	XX	1X	XX

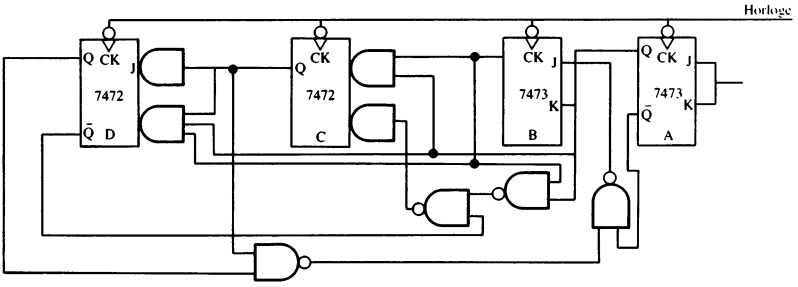
J_D, K_D

Q_DQ_C \ Q_BQ_A	00	01	11	10
00	0X	0X	0X	0X
01	1X	XX	XX	XX
11	X0	X0	X1	X0
10	XX	XX	X0	XX

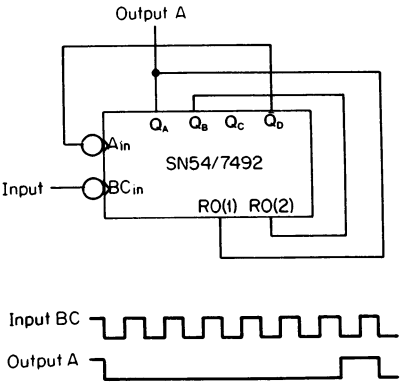
$$J_C = Q_AQ_B \qquad J_D = Q_C$$

$$K_C = Q_AQ_B + \overline{Q_D} \qquad K_D = Q_AQ_BQ_C$$

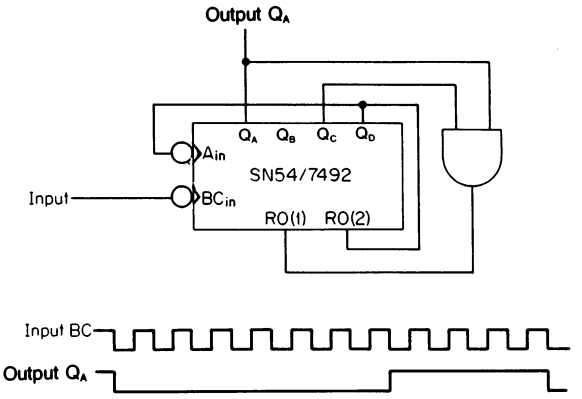
$$= \overline{\overline{Q_AQ_B} \cdot Q_D}$$



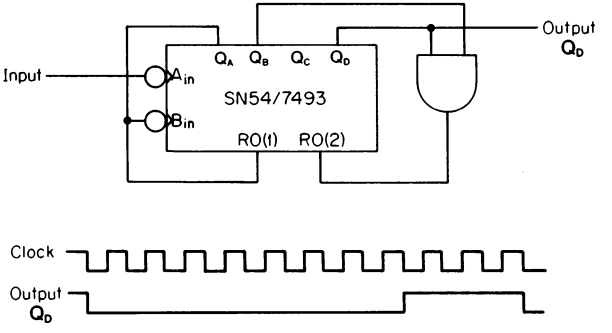
5-23



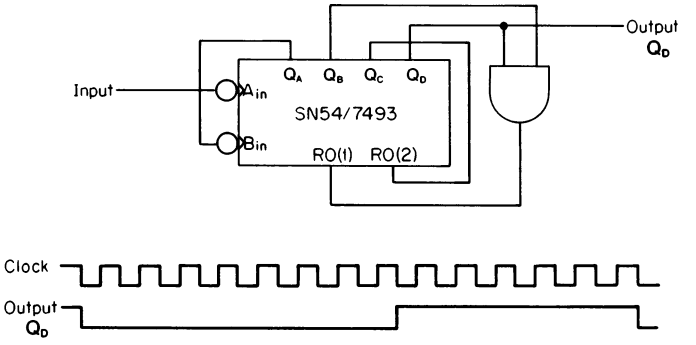
5-25


Output Q_A
Q_A Q_B Q_C Q_D
A_{in}
SN54/7492
Input
BC_{in}
RO(1) RO(2)
Input BC
Output Q_A

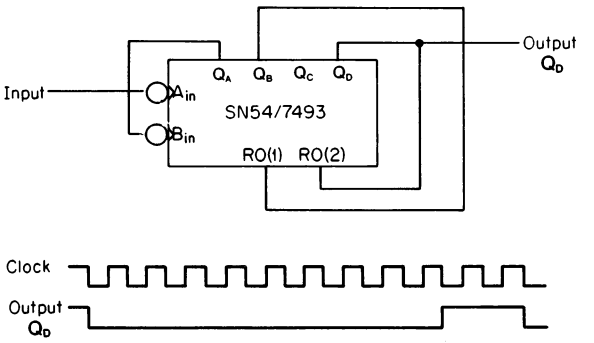
5-27


Output Q_D
Q_A Q_B Q_C Q_D
Input
A_{in}
SN54/7493
B_{in}
RO(1) RO(2)
Clock
Output Q_D

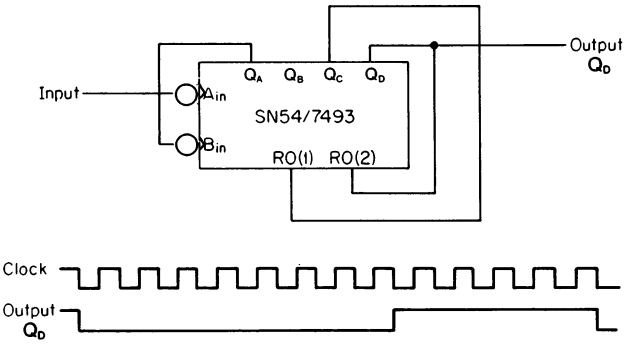
5-29


Output Q_D
Q_A Q_B Q_C Q_D
Input
A_{in}
SN54/7493
B_{in}
RO(1) RO(2)
Clock
Output Q_D

5-31



5-33



5-35

0	0	0
1	0	0
1	1	0
1	1	1
0	1	1
0	0	1

5-37

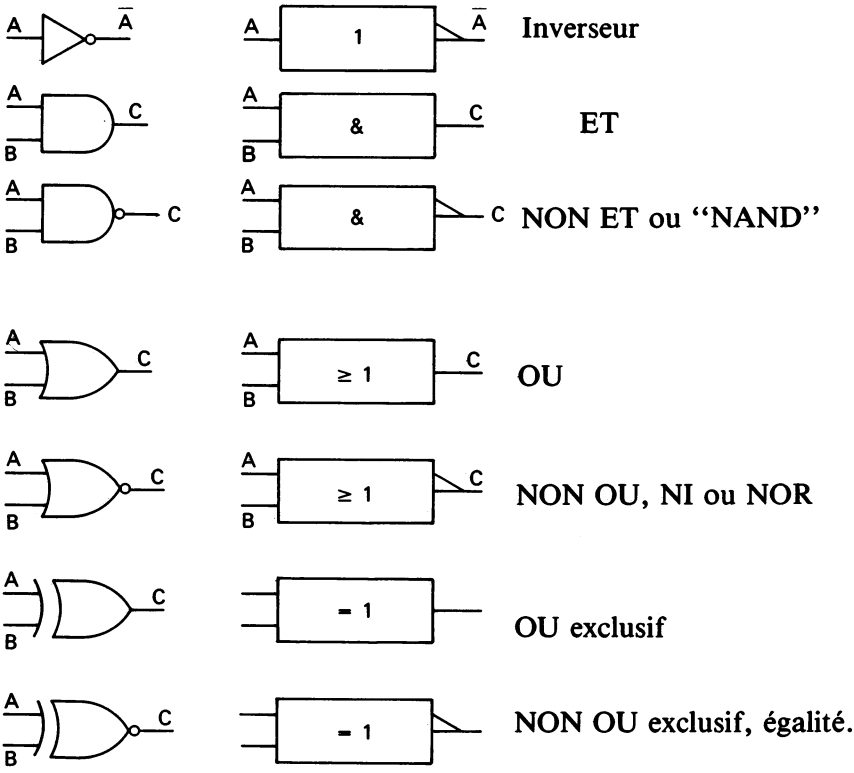
1	0	1	1
0	1	0	1
0	0	1	0
→ 0	0	0	1
0	0	0	0
1	0	0	0
1	1	0	0
1	1	1	0
1	1	1	1
0	1	1	1
0	0	1	1
0	0	0	1

Appendice C

Représentation CEI des circuits logiques

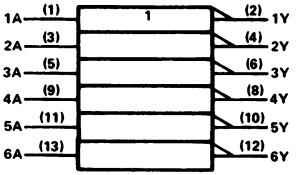
Une nouvelle forme de représentation des circuits logiques a été adoptée par la Commission électrotechnique internationale (CEI ou IEC en américain). Cette nouvelle représentation simplifie les schémas et permet de faire des dessins à des échelles différentes sans difficultés. Elle présente toutefois l'inconvénient d'être moins "parlante" au premier coup d'oeil.

Voici la représentation de quelques fonctions élémentaires

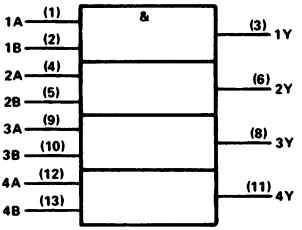


Voici les nouveaux symboles logiques des principaux circuits intégrés mentionnés dans le volume.

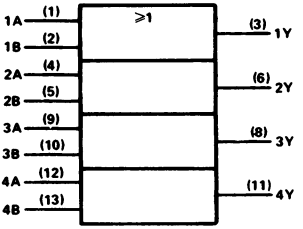
7404 Inverseur



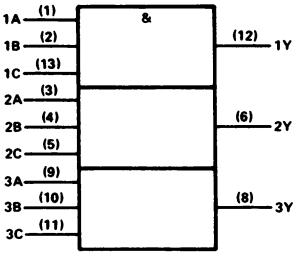
7408 Portes ET à 2 entrées



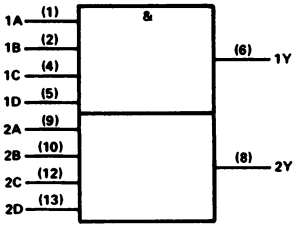
7432 Portes OU à 2 entrées



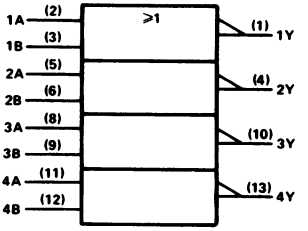
7411 Portes ET à 3 entrées



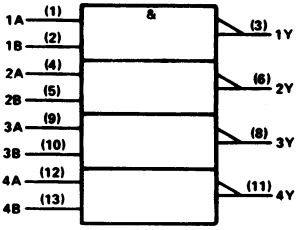
7421 Portes ET à 4 entrées



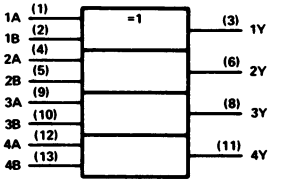
7402 Portes NON OU



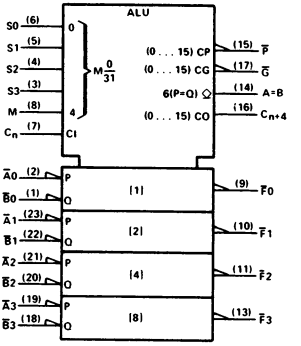
7400 Portes NON ET



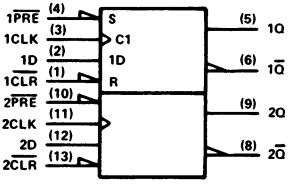
7486 Portes OU exclusif



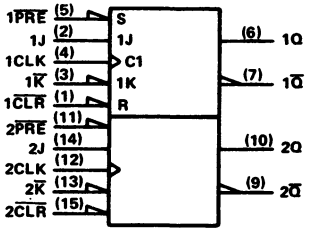
74181 Unité arithmétique et logique



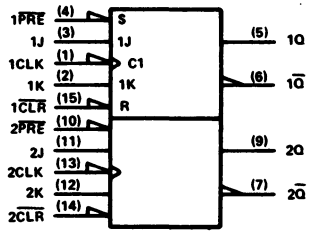
7474 Bascule de type D



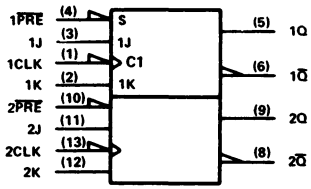
74109 Bascule JK



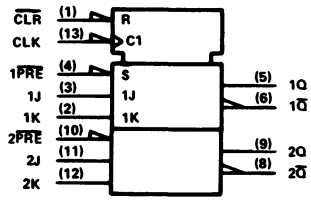
74112 Bascule JK



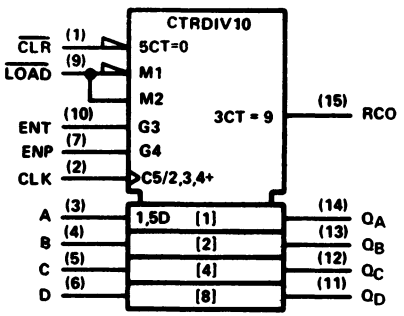
74113 Bascule JK



74114 Bascule JK

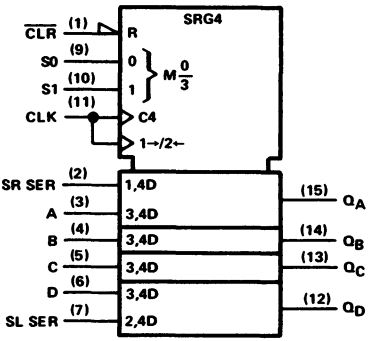


74160 Compteur synchrome



74194

Registre à décalage



Source: *The TTL Data Book*, volume 3, Texas Instruments, 1984.

Appendice D

Index des schémas de circuits intégrés

C.I. N°	PAGE	C.I. N°	PAGE	C.I. N°	PAGE
7400	62	7471	191	74112	196
7402	62	7472	191	74113	196
7404	48-49	7473	191	74114	196
7408	49-50	7474	192	74121	197
7410	104	7475	182	74122	197
7411	54-105	7476	192	74123	197
7420	104	7478	192	74138	157
7421	54-105	7483	126-136	74150	160
7427	105	7486	66	74151	161
7430	104	7487	131-132-136	74152	162
7432	50-51	7490	215-216	74154	158-163
7442	164	7492	215-216	74155	156
7443	165	7493	215-216	74156	156
7444	165	7496	228	74160	217 à 221
7446	171-173	74101	193	74161	217 à 221
7447	171-173	74102	193	74162	217 à 221
7448	171-172-173	74103	193	74163	217 à 221
7449	171-172-173	74106	194	74181	147-153
7450	112	74107	194	74194	226
7451	105-106	74108	194	74198	229-231
7454	106	74109	195	74199	230-231
7460	112	74110	195	74266	67
7470	190	74111	195	74279	182

LEXIQUE

A
Active high data: données actives de niveau haut
Active low data: données actives du niveau bas
Address: adresse
ALU (Arithmetic and Logic Unit): unité arithmétique et logique
AND: ET
Arithmetic operation: opération arithmétique
Asynchronous: asynchrone

B
BCD (Binary Coded Decimal): décimal codé binaire
Bidirectional: bidirectionnel
Binary: binaire
Biquinary: biquinaire
Bistable: bascule
Bit (Binary Digit): bit
Blanking: effacement
Buffer: tampon

C
Carry: report (addition) — retenue (soustraction)
Carry generate output: sortie du report généré
Carry Input: entrée du report
Carry propagate output: sortie de propagation du report
CK (clock): horloge (T)
CLR (Clear): effacement, remise à zéro
Clear (to): effacer, remettre à zéro
Common: commun
Comparator: comparateur
Connect (to): connecter
Control: commande
Control (to): commander
Count (to): compter
Count Sequence: séquence de comptage
Counter: compteur

D
Data: données
Data lockout: verrouillage des données
Decimal: décimal

Decoder: décodeur
Decade counter: compteur décimal
Demultiplexer: démultiplexeur
Designation: désignation
Disable (to): invalider, mettre hors service
Display: affichage, visuel
Divide-by-twelve counter: compteur modulo 12
Dual: deux, double
Dynamic: dynamique

E
Edge triggered: à déclenchement sur front d'impulsion
Enable (to): valider, permettre
Equivalent: équivalent
Excess three: plus trois

F
Flip-flop: bascule
4-bit binary counter: compteur hexadécimal
Full: complet
Function: fonction
Function table: table de fonctions
Functional bloc diagram: diagramme synoptique des fonctions
Functions outputs: sorties des fonctions
Function select input: entrées de sélection des fonctions

G
Gate: porte
Gated: à porte (s) [ET, ...]
GND (ground): masse
Ground (to): mettre à la masse

H
H (High ou high level): niveau haut
H (High speed): à vitesse rapide

I
Identification: identification
Inhibit (to): interdire, invalider
Inderteminate: indéterminé
Input: entrée
Invalid: incorrect

Invert (to): inverser, complémenter
Irrelevant: indifférent

J

K

L
L (Low consumption): faible consommation
L (Low or low level): niveau bas
Lamp test: essai de la lampe
Latch: bascule, verrou
Left: gauche
Level: niveau
Line: ligne
Load (to): charger
Load: chargement
Logic function: fonction logique
Look-ahead: lecture par anticipation
LSI (Large Scale Integration: intégration à grande échelle

M
Master-slave flip-flop: bascule maître-esclave
Minus: moins
Monostable multivibrator: multivibrateur
Mode control input: entrée du mode de commande
MSI (Medium Scale Integration Monostable): intégration à moyenne échelle

N
NAND: NON-ET
NC (No Connection): aucune connexion
Negative edge triggered: à basculement sur front descendant d'impulsion
Negative logic: logique négative
No external connection: aucune connexion externe
NOR: NON-OU
Numerical: numérique

O
OFF: hors-circuit
ON: en circuit
OR: OU
Output: sortie

P
Package: boîtier
Parallel: parallèle
Pin: broche
Pin designation: affectation des broches
Positive edge triggered: à basculement sur front montant d'impulsion
Positive logic: logique positive
PR (PRESET): remise à 1

Q

R
R (RESET): remise à zéro
Register: registre
Resultant: résultant
Right: droit
Ripple: cascade
Ripple carry output: sortie du report en cascade

S
S (Set): remise à 1
Schematics: schéma
Segment: segment
Select: sélection, adresse
Sequence: séquence
Serial: série
Shift: décalage
SSI (Small Scale Integration): intégration à petite échelle
State: état
Steady: stable
Strobe: échantillonnage
Supply voltage: tension d'alimentation
Synchronous: synchrone

T
T (Toggle): basculement, signal d'horloge
Timing: chronogramme, synchronisation
Toggle (to): basculer
Trigger: déclencheur, bascule
Trigger (to): déclencher, basculer
Transition: transition
TTL (Transistor Transistor Logic): logique transistor, transistor

U
Universal: universel

V

W
Word A inputs: entrées du mot A

X
X (irrelevant): indifférent

Y

Z

INDEX

RETRONIK.FR 2023

0-07-548985-6