UNIVERSITÉ DE NEUCHATEL
INSTITUT DE MICROTECHNIQUE

Description des systèmes logiques combinatoires par des fonctions booléennes caractéristiques permettant le calcul des impliquants premiers et des couvertures irrédondantes au moyen d'un algorithme unique adapté aux petits ordinateurs

THÈSE

PRÉSENTÉE À LA FACULTÉ DES SCIENCES
POUR OBTENIR LE GRADE DE DOCTEUR ÈS SCIENCES

PAR

J.-J. MONBARON

IMPRIMERIE DE L'OUEST SA, 2034 PESEUX
1980

IMPRIMATUR POUR LA THÈSE

Description des systèmes logiques combinatoires par des fonctions booléennes caractéristiques permettant le calcul des impliquants premiers et des couvertures irrédondantes au moyen d'un algorithme unique adapté aux petits ordinateurs.

de M. Jean-Jacques Monbaron

UNIVERSITÉ DE NEUCHATEL

FACULTÉ DES SCIENCES

La Faculté des sciences de l'Université de Neuchâtel, sur le rapport des membres du jury,

Messieurs F. Pellandini, A. Robert, A. Shah et J. Zahnd (EPF Lausanne)

autorise l'impression de la présente thèse.

Neuchâtel, le 27 mars 1980

Le doyen :

K. Bo[illegible]

TABLE DES MATIERES.

INTRODUCTION.

L'électronique a connu un développement très rapide ces vingts dernières années. Un fait remarquable de cette discipline est qu'elle s'est divisée en deux branches distinctes : l'électronique analogique et l'électronique digitale. Alors que l'électronique analogique pouvait se développer par des méthodes de description et de calcul très répandues et proches de celles de la physique classique, telles que le calcul différentiel et intégral, la théorie des fonctions complexes, le calcul matriciel, etc..., l'électronique digitale a du avoir recours à des méthodes de calcul nouvelles, basées sur l'algèbre de Boole, la théorie des corps de Le Galois, pour aboutir aux notions modernes de théorie des automates, programmation structurée, etc. La rapidité de cette évolution a donné lieu à certaines lacunes dans les formalismes de description et des méthodes de calcul, qui souffrent encore d'un manque d'unification. Il est à cet égard intéressant de constater que les récents travaux de Menger (14) sur les transformées de Fourier des systèmes logiques, de Thayse et Davio (15) sur le calcul différentiel booléen tendent à rapprocher les méthodes de l'électronique digitale de celles de la physique.

L'influence de ces lacunes ne se fait que peu ressentir lorsqu'on réalise une machine digitale en peu d'exemplaires à l'aide de circuits intégrés du commerce, car bien qu'hautement souhaitable, la mise en oeuvre de méthodes systématiques se révèle souvent trop coûteuse en regard des méthodes heurystiques. Ceci explique que les premières "machines à simplifier", telles que celle de Florine (8), celle de Mange (9), celle de Parkhomenko (10), sont restées peu connues des ingénieurs concepteurs.

Le cas actuel de l'intégration de systèmes complexes sur une seule puce est fort différent. Ces circuits sont destinés à être produits en très grand nombre. Ainsi chaque réduction de la complexité, même faible, peut amener une amélioration non négligeable de la fiabilité, du rendement de la fabrication, de la surface ou encore de la consommation.

De nombreux auteurs (6) (11) (12) (16) (17) ..., ont exposé des méthodes et des programmes de synthèse sur ordinateurs. Ces programmes permettent d'effectuer la synthèse des systèmes logiques combinatoires en formes normales, minimales ou irrédondantes, afin d'optimiser selon certains critères de coût, des circuits tels que les décodeurs, les PLA (programmable logic arrays), les unités arithmétiques et logiques, sur des ordinateurs conventionnels.

Le but de ce travail est de fournir un formalisme et une méthode de synthèse en formes normales minimales ou irrédondantes des systèmes logiques combinatoires, programmable sur de très petites machines, telles que les calculatrices de table, les systèmes à microprocesseurs ou les miniordinateurs, afin de rendre accessible l'optimisation automatique des systèmes à un plus grand nombre d'utilisateurs.

Pour ce faire, il a été développé sur la base d'un précédent travail (20) un programme de calcul des impliquants premiers pour des systèmes pouvant comprendre jusqu'à plusieurs milliers d'entrées et de sorties, selon une méthode proche de celle de Kazakof (5). Cette méthode consistait à former dans l'ordre croissant de complexité, tous les produits possibles des variables apparaissant dans le problème et à tester leur appartenance aux différentes fonctions de sortie du système. Chaque produit valide formé étant premier, on pouvait s'interdire de former les termes de coût plus élevés inclus. Le processus pouvait donc s'arrêter dès qu'on avait obtenu un ensemble de termes satisfaisant le problème. L'idée de cette méthode reposait sur le fait que les systèmes dépendant d'un grand nombre de variables sont toujours fortement indéfinis, ce qui permet de ne former qu'un petit nombre de produits,

qu'on rejette s'ils contiennent un zéro de la fonction à synthétiser. Cette manière de faire est dans son principe identique à la démarche de Tison (18) pour la recherche des consensus entre plusieurs monomes (produits). Cette recherche consiste en effet à former tous les produits possibles des variables monoformes (c'est-à-dire non nuls) d'une implication. Chaque produit formé est alors impliquant si la somme des résidus biformes est une égalité à 1, ce qui revient à tester que le produit formé ne contient pas de zéro de la fonction ! L'inconvénient de la recherche des consensus réside cependant dans le fait que pour une fonction contenant au départ beaucoup d'impliquants, (même si ceux-ci sont des états indéfinis), le nombre de produits à former devient trop grand. Il est cependant intéressant de constater que l'utilisation d'une table de Karnaugh procède toujours du même principe :
On forme les plus grandes surfaces possibles (représentant des produits de variables) ne contenant pas de zéros, en s'évitant par là de former des impliquants inclus à d'autres. Cette méthode est cependant aussi limitée par le nombre de variables, même si le problème est fortement indéfini.

L'algorithme exposé dans ce travail se ramène aux processus précédents par le raisonnement suivant :

Puisque le problème consiste à trouver tous les produits possibles ne contenant pas de zéros, il suffit d'enlever à la fonction contenant tous les produits possibles, c'est-à-dire la fonction booléenne 1, la fonction Z contenant tous les zéros du problème, en calculant l'expression booléenne $F = 1 \cdot \bar{Z}$. Ceci est le principe du "sharp algorithm", dont une version originale pour le traitement des systèmes à plusieurs sorties est présentée. Cette version permet le calcul des impliquants premiers d'un système dépendant d'un grand nombre de variables, grâce à une méthode de codage consistant à ne mémoriser que les états définis des variables (d'entrée et de sortie), ce qui représente une grande économie de mémoires et de temps de calcul pour les petites machines. En effet, on peut montrer que le nombre de produits à former dépend linéairement du nombre d'états définis des variables du problème, lequel représente le contenu informationnel du système à traiter.

Dans sa théorie des consensus, Tison (18) a donné une solution élégante au calcul des couvertures irrédondantes et minimales d'une fonction booléenne, en définissant une fonction "base" qui dépend uniquement de variables booléennes auxiliaires, liées à la présence ou à l'absence de chaque impliquant premier dans une couverture de la fonction à réaliser.

Il a montré que cette fonction s'obtient par le calcul des relations de consensus de chaque impliquant premier avec tous les autres. Une version générale de cette méthode sera présentée, consistant simplement à calculer les solutions d'une équation F(X) = 1, représentées par les impliquants premiers de celle-ci. Ainsi un seul et même algorithme sera utilisé à la fois pour le calcul des impliquants premiers et des couvertures irrédondantes d'un système de plusieurs fonctions.

CHAPITRE 1. CALCUL DES IMPLIQUANTS PREMIERS D'UNE FONCTION SIMPLE.

1.1. Introduction.

Le but de ce chapitre est de montrer qu'une réalisation "optimale", d'un système logique à 2 niveaux de portes, passe par le calcul des impliquants premiers de la fonction décrivant le système.

Le critère d'optimisation sera clarifié par la présentation des notions d'impliquants, d'impliquants premiers, de somme irrédondante.

Un procédé systématique de calcul des impliquants premiers d'une fonction simple est démontré.

1.2. Rappel de formules de l'algèbre de Boole.

Nous donnons sans démonstration un recueil de formules (axiomes et propriétés) qui seront utilisées dans la suite du travail.

Définition :

On appelle algèbre de Boole un ensemble A muni d'une opération X + Y (somme de deux éléments de A), d'une opération $X\underline{Y}$ (produit de deux éléments de A), d'une opération $\bar{X}$ (complémentation d'un élément de A), d'un élément 0, d'un élément 1, tels que les axiomes suivants soient satisfaits :

1. $X + Y = Y + X$	$XY = YX$
2. $X + (Y + Z) = (X + Y) + Z$	$X(YZ) = (XY)Z$
3. $X + (YZ) = (X + Y)(X + Z)$	$X(Y + Z) = (XY) + (XZ)$
4. $X + 0 = X$	$X1 = X$
5. $X + \bar{X} = 1$	$X\bar{X} = 0$

Théorème : Les formules suivantes sont des conséquences des axiomes (1) à (5).

6. $X + X = X$	$XX = X$
7. $X + 1 = 1$	$X0 = 0$
8. $X + (XY) = X$	$X(X + Y) = X$
9. $X + Y = X + (\bar{X}Y)$	$XY = X(\bar{X} + Y)$
10. $X = (XY) + (X\bar{Y})$	$X = (X + Y)(X + \bar{Y})$
11. $X + Y = Y \iff XY = X$	
12. $X + Y = 0 \iff (X = 0$ et $Y = 0)$	$XY = 1 \iff (X = 1$ et $Y = 1)$
13. $Y = \bar{X} \iff (X + Y = 1$ et $XY = 0)$	
14. $\bar{\bar{X}} = X$	
15. $X = Y \iff \bar{X} = \bar{Y}$	
16. $\overline{X + Y} = \bar{X}\bar{Y}$	$\overline{XY} = \bar{X} + \bar{Y}$.

On trouve une démonstration de ces formules à partir des axiomes (1) à (5) dans Harrison par exemple.

Définition : Dans toute algèbre de Boole A, la relation $X \leq Y$ entre deux éléments de A est définie par :

17. $X \leq Y \Leftrightarrow X + Y = Y$

Il en découle par (11) :

18. $X \leq Y \Leftrightarrow XY = X$

Théorème : On démontre les formules suivantes à partir de ce qui précède :

19. $X \leq X$

20. $(X \leq Y \text{ et } Y \leq X) \Leftrightarrow X = Y$

21. $(X \leq Y \text{ et } Y \leq Z) \Rightarrow X \leq Z$

22. $0 \leq Y \leq 1$

23. $X \leq Y \Leftrightarrow \bar{X} + Y = 1$

24. $X \leq Y \Leftrightarrow X\bar{Y} = 0$

25. $X \leq Y \Leftrightarrow \bar{Y} \leq \bar{X}$

26. $(X \leq Y \text{ et } X' \leq Y') \Rightarrow X + X' \leq Y + Y'$

27. $(X \leq Y \text{ et } X' \leq Y') \Rightarrow XX' \leq YY'$

28. $(X \leq Y \text{ et } X' \leq Y) \Rightarrow X + X' \leq Y$

29. $(X \leq Y \text{ et } X \leq Y') \Rightarrow X \leq YY'$

30. $X \leq X + Y$

31. $XY \leq X$.

Théorème : Si X,Y sont deux éléments d'une algèbre de Boole, alors :

32. $X = Y \Leftrightarrow (\bar{X} + Y)(\bar{Y} + X) = 1$

33. $X = Y \Leftrightarrow X\bar{Y} + Y\bar{X} = 0$

Démonstration par (20) et (23), ou (20) et (24), et (12)

Commentaire : Toutes les formules énoncées jusqu'ici sont supposées familières au lecteur, et seront utilisées en général sans qu'elles soient mentionnées.

1.3. Notion d'impliquant.

La notion d'impliquant est liée à une réalisation "à 2 niveaux", typique des circuits logiques actuels. En effet, on décrit le fonctionnement d'un système le plus souvent par la somme des configurations dans lesquelles il doit avoir une action donnée.

Cette somme (réunion) booléenne s'écrit par exemple :

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

On peut dire que le fait que $a\bar{b} = 1$ implique que $F = 1$, d'où le terme "impliquant".

Définition : soit $F(a_1, a_2, \ldots, a_N)$ une fonction booléenne et $A(a_1, a_2, \ldots, a_N)$ un produit de variables, alors on dit que A est impliquant de F si :

$$AF = A$$

Cette relation s'écrit aussi :

$$A \leqslant F$$

ou encore :

$$A \Longrightarrow F$$

1.4. Notion d'impliquant premier.

Soient : $F(a_1,a_2,\ldots,a_N)$ et $A(a_1,a_2,\ldots,a_N)$

une fonction et un produit de variables, on dit que A est impliquant premier de F s'il n'existe aucun autre impliquant $B(a_1,a_2,\ldots,a_N)$ de F tel que :

$$A \leq B$$

Pratiquement un impliquant premier est un produit composé d'un minimum de variables, "couvrant" le plus grand nombre d'états possibles.

Exemple : si $a\bar{c}$ est premier

alors $a\bar{b}\bar{c}$ n'est pas premier.

1.5. Somme de produits irrédondante.

La notion de forme "minimale" d'une fonction à 2 niveaux implique celle de forme "irrédondante".

Définition : Une somme de produits est irrédondante si on ne peut enlever aucun terme de la somme ni aucune variable dans chaque produit sans changer la valeur de la somme.

Exemple : La fonction $F = a_1a_2 + \bar{a}_2a_3$ est irrédondante.

La fonction $F = a_1 + a_1a_2$ n'est pas

irrédondante car on peut enlever le terme $a_1 a_2$ sans changer la valeur de F.

La fonction $F = a_1 + \bar{a}_1 a_2$ n'est pas irrédondante car on peut écrire $F = a_1 + a_2$.

La fonction $F = a_1 a_2 + a_1 \bar{a}_2$ n'est pas irrédondante car on peut écrire $F = a_1$.

Théorème 1 : Toute somme irrédondante n'est composée que d'impliquants premiers.

Démonstration :

1) Soit $F(a_1, a_2, \ldots, a_N) = A_1 + A_2 + \ldots + A_i + \ldots + A_N$

avec : $A_i = A_i(a_1, a_2, \ldots, a_N)$ pour tout i.

Tout terme A_i non premier est de la forme $B_i \cdot X$ avec B_i impliquant premier.
En effet, la relation $B_i \cdot X \leq B_i$ est toujours vraie !

2) On peut donc remplacer tout A_i non premier par l'impliquant premier qui le contient, ce qui consiste à diminuer le nombre de lettres d'un terme.

3) Si 2 termes A_i et A_j sont remplacés par le même impliquant premier, on n'écrit celui-ci qu'une fois, ce qui revient à supprimer un terme de la somme.

On peut conclure que toute somme irrédondante n'est composée que d'impliquants premiers !

1.6. Processus de calcul des impliquants premiers.

Le principe du calcul des impliquants premiers d'une fonction F en somme de produits est simple :

il suffit de transformer par De Morgan $\bar{F}$ (somme de produits) en F (produits de sommes), chaque variable étant inversée, et d'effectuer le calcul pour obtenir F en somme de produits parmi lesquels se trouvent tous les impliquants premiers.

1.6.1. Calcul des impliquants premiers d'un produit de fonctions.

Ce théorème préalable sera utilisé pour la démonstration par récurrence du § 1.6.2.

Théorème 2 : Soient G et H, 2 fonctions exprimées par la somme de tous leurs impliquants premiers. Les impliquants premiers de F = GH s'obtiennent en effectuant le produit des impliquants premiers de G et de H.

Démonstration :

1) Supposons A tel que : $A \leq F$ A premier

2) $A \leq F \Longrightarrow A \leq GH \Longrightarrow A \leq G$

(L'inclusion est transitive)

De même : $A \leq GH \Longrightarrow A \leq H$

3) Prenons un impliquant premier B de H tel que $A \leqslant B$,

alors : $A \leqslant B \leqslant H$

et un impliquant premier de C de G tel que $A \leqslant C$,

alors : $A \leqslant C \leqslant G$

4) Il s'ensuit que :

$$A \leqslant BC \leqslant GH = F$$

5) Comme A est premier, on doit avoir

$$A = BC$$

Conclusion : A = BC a été obtenu en effectuant le produit GH.

Exemple :

$G = i + jk$ $\qquad$ $H = im + no$

$F = GH = im + ino + ijkm + jkno$

Remarque 1 : Tous les produits formés par le processus de multiplication ne sont pas forcément premiers.

En effet, si : BC = B, c'est-à-dire $B \leqslant C$, B est impliquant premier de F = GH et tout produit BD avec $D \leqslant G$ est inférieur ou égal à B, donc non premier, ou bien égal à B.

Dans l'exemple qui précède, on a bien

$$i \gg im$$

le produit partiel $i \cdot im = im$ étant premier, l'autre produit formé à partir de im, tel que ijkm est non premier.

Théorème 3 : Une fonction logique différente de 1 composée d'une somme de variables (sous forme directe ou complémentée), est représentée par la somme de ses impliquants premiers.

Démonstration :

1) Soit $F = a_1 + a_2 + \ldots + a_n$

avec a_i variable sous forme directe ou complémentée.

2) Supposons a_i non premier.

Il doit alors exister un impliquant premier X tel que $a_i < X$.

3) La seule solution est X = 1, ce qui est exclu dans l'énoncé du théorème.

4) a_i est donc premier.

En effet :

$F = x_1 + \bar{x}_1 + x_2$ n'est pas une somme d'impliquants premiers car on peut écrire :

$$F = 1 + x_2 = 1$$

1.6.2. Algorithme d'inversion d'une fonction.

L'inversion d'une fonction F par De Morgan suivie du calcul de l'expression en somme de produits fournit tous les impliquants premiers de $\bar{F}$.

Soit une fonction F donnée sous la forme d'une somme de produits :

$$F = A_1 + A_2 + A_3 + \ldots + A_n$$

Par transformation de De Morgan on peut écrire $\bar{F}$ comme produit de sommes :

$$\bar{F} = \bar{A}_1 \cdot \bar{A}_2 \cdot \bar{A}_3 \cdot \ldots \cdot \bar{A}_n$$

Chaque $\bar{A}_i$ est la somme des inverses des variables qui composent A_i.

Si aucun A_i n'est nul, chaque $\bar{A}_i$ est une somme d'impliquants premiers. (cf. théorème 3, chap. 1.6.1., ci-dessus).

En posant :

$P_0 = 1$ P_0 est somme d'impliquants premiers...

$P_1 = P_0 \cdot \bar{A}_1$ En développant P_1 en somme de produits, on obtient P_1 sous forme de somme de ses impliquants premiers (en vertu du théorème 2, chap. 1.6.1.)

En poursuivant le processus itérativement jusqu'à

$$P_n = P_{n-1} \cdot \bar{A}_n$$

on obtient $\bar{F} = P_n$ sous forme de somme qui contient tous ses impliquants premiers !

Cet algorithme est appelé aussi sharp algorithm ou extraction algorithm dans la littérature de langue anglaise. Tison l'appelle méthode de passage à la duale.

Remarque 2 : Si on dispose de F au départ (sous une forme de somme de produits), il faut d'abord exprimer $\bar{F}$ sous forme de somme de produits au moyen de l'algorithme précité, puis par une seconde application du processus, on obtient F sous forme de somme de tous ses impliquants premiers.

Exemple :

Soit : $F = ab + \bar{b}c$

Par De Morgan on peut écrire :

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

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

On a obtenu les impliquants premiers de $\bar{F}$.

Calculons les impliquants premiers de F par le même procédé :

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

On calcule d'abord l'expression :

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

Les deux termes non premiers sont éliminés, puis :

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

On constate que l'impliquant premier ac ne figurant pas dans la donnée est apparu. Cet exemple est un cas particulier du consensus.

Remarque 3 : Si au lieu de prendre $P_0 = 1$ au début de l'itération, on pose $P_0 = G(a_1, a_2, \ldots, a_n)$, où G est une fonction arbitraire, l'algorithme d'inversion permet de calculer l'expression $G\bar{F}$.

Il est ainsi possible de calculer une expression de fonctions booléennes quelconques, à condition de transformer celle-ci de manière à ce qu'elle ne contienne plus que l'opérateur $G\bar{F}$!

Exemple :

Calculer l'expression $Z = \bar{G} + F$

On peut se ramener à : $Z = \overline{(G\bar{F})}$

Ainsi 2 applications du processus suffisent pour obtenir Z sous forme de somme de ses impliquants premiers.

1.6.3. Exemple No 2.

Soit à calculer les impliquants premiers de :

$$F = \bar{a}\bar{b}c\bar{d} + \bar{a}\bar{b}cd + \bar{a}b\bar{c}d + \bar{a}bc\bar{d} + \bar{a}bcd + a\bar{b}\bar{c}\bar{d} + ab\bar{c}\bar{d} + abc\bar{d}$$

On procède d'abord au calcul de $\bar{F}$, selon l'algorithme 1.6.2.

$$P_0 = 1$$

$$P_1 = P_0(a + b + \bar{c} + d) = a + b + \bar{c} + d$$

$$P_2 = P_1(a + b + \bar{c} + \bar{d}) = a + b + \bar{c}$$

$$P_3 = P_2(a + \bar{b} + c + \bar{d}) = a + bc + b\bar{d} + \bar{b}\bar{c} + \bar{c}\bar{d}$$

$$P_4 = P_3(a + \bar{b} + \bar{c} + d) = a + bcd + b\cancel{c\bar{d}} + \bar{b}\bar{c} + \bar{c}\bar{d}$$

$$P_5 = P_4(a + \bar{b} + \bar{c} + \bar{d}) = a + \bar{b}\bar{c} + \bar{c}\bar{d}$$

$$P_6 = P_5(\bar{a} + b + c + d) = ab + ac + ad + \bar{a}\bar{b}\bar{c} + \bar{b}cd + \bar{a}\bar{c}\bar{d} + b\bar{c}\bar{d}$$

$$P_7 = P_6(\bar{a} + \bar{b} + c + d) = \cancel{abc} + \cancel{abd} + ac + ad + \bar{a}\bar{b}\bar{c} + \bar{b}cd + \bar{a}\bar{c}\bar{d} + \cancel{\bar{a}bc\bar{d}}$$

$$\bar{F} = P_8 = P_7(\bar{a} + \bar{b} + c + \bar{d}) = \cancel{abcd} + ac + a\bar{b}d + \bar{a}\bar{b}\bar{c} + \bar{b}cd + \bar{a}\bar{c}\bar{d}$$

On a obtenu $\bar{F}$ sous forme de somme d'impliquants premiers, on inverse alors $\bar{F}$ pour obtenir F de la même manière :

$$P_0 = 1$$

$$P_1 = \bar{a} + \bar{c}$$

$$P_2 = P_1(\bar{a} + b + \bar{d}) = \bar{a} + b\bar{c} + \bar{c}\bar{d}$$

$$P_3 = P_2(a + b + c) = \bar{a}b + \bar{a}c + b\bar{c} + a\bar{c}\bar{d} + \cancel{b\bar{c}d}$$

$$P_4 = P_3(b + c + \bar{d}) = \bar{a}b + \bar{a}c + b\bar{c} + a\bar{c}\bar{d}$$

$$F = P_5 = P_4(a + c + d) = \cancel{\bar{a}bc} + \bar{a}bd + \bar{a}c + ab\bar{c} + b\bar{c}d + a\bar{c}\bar{d}$$

Cet exemple montre que l'algorithme d'inversion ne requiert que la mise en oeuvre du produit booléen (y-compris tests d'inclusion). Il reste donc très simple à réaliser sur de petites machines, telles que calculatrice de bureau, microprocesseur, etc...

De plus, nous verrons au chapitre 2.9. que chaque fois que F est donnée par un tableau de vérité, le calcul des impliquants premiers de F s'effectue en une seule inversion.

CHAPITRE 2. DESCRIPTION D'UNE FONCTION INCOMPLETEMENT DEFINIE.

Introduction.

Nous allons dans ce chapitre étudier directement la synthèse en somme de produit irrédondante pour des fonctions imcomplètement définies, le cas des fonctions complètement définies apparaissant comme un cas particulier du précédent.

2.1. Fonction véracité d'une équation booléenne.

Soit $F(X_1, X_2, \ldots, X_n) = G(X_1, X_2, \ldots, X_n)$

L'équation F = G est équivalente à :

$$F \leqslant G \quad \text{et} \quad G \leqslant F$$

C'est-à-dire que toute solution du système d'équations est solution de l'équation et réciproquement.

Or : $F \leqslant G \iff \bar{F} + G = 1$

$G \leqslant F \iff \bar{G} + F = 1$

Donc les solutions du système de l'équation F = G sont les solutions du système d'équations :

$$\bar{F} + G = 1$$

$$\bar{G} + F = 1$$

et il est évident que les solutions du système sont aussi solutions de :

$$(\bar{F} + G)(\bar{G} + F) = 1$$

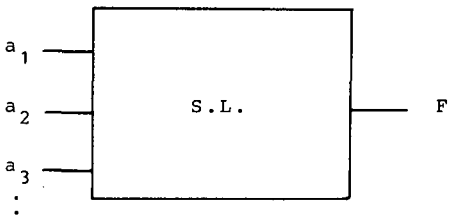
La fonction :

$$V(F = G) = FG + \bar{F}\bar{G}$$

est appelée fonction véracité, ou fonction caractéristique de l'équation booléenne F = G.

Cette fonction prend la valeur 1 lorsque F = G, et 0 sinon.

2.2. Description d'un système logique avec états indéfinis.



Décrivons le système en se donnant pour certains états des entrées, les états de la sortie à l'aide d'un tableau de correspondances.

Par ex. :

a_1	a_2	a_3	F
0	0	1	1
1	0	0	0
1	0	1	1
1	1	1	0

Le problème consiste à décrire le comportement du système à l'aide d'une fonction booléenne.

Attribuons à chaque entrée une variable booléenne caractéristique a_1, a_2, a_3, ...

Notons par x_{ij} la valeur que prend l'entrée a_i dans la ligne j du tableau.

La fonction véracité de ce fait s'écrit :

$$v(a_i = x_{ij})$$

ou x_{ij} prend la valeur 0 ou 1.

2.3. Fonction véracité d'un état des entrées.

Nous avons vu au § 2.2. que la fonction véracité de l'état d'une entrée caractérisée par a_i s'écrit :

$$v(a_i = x_{ij})$$ pour la ligne j du tableau.

Nous pouvons maintenant écrire la fonction véracité de cette égalité à l'aide de l'expression du chap. 2.1. :

$$v(a_i = x_{ij}) = a_i \cdot x_{ij} + (\bar{a}_i) \cdot (\bar{x}_{ij})$$

Si $x_{ij} = 0$ il vient :

$$v(a_i = x_{ij}) = a_i \cdot 0 + (\bar{a}_i) \cdot 1 = (\bar{a}_i)$$

Si $x_{ij} = 1$ il vient :

$$v(a_i = x_{ij}) = a_i \cdot 1 + (\bar{a}_i) \cdot 0 = a_i$$

De même, le fait que plusieurs entrées a_1, a_2, a_3,... prennent simultanément une valeur x_{ij}, y_{ij}, z_{ij} se décrit par le produit des fonctions véracité liées à chaque entrée.

Par exemple :

$$v(a_1 = 1 \text{ et } a_2 = 0 \text{ et } a_3 = 1) = a_1\bar{a}_2a_3$$

L'expression $a_1\bar{a}_2a_3$ représente une "forme codée" de la valeur 101 des variables d'entrées a_1, a_2 et a_3 de la 3e ligne du tableau.

2.4. Fonction véracité d'un état de la sortie.

Comme pour les états d'entrée, on peut définir les

états de sortie par une fonction véracité reliant la variable F qui identifie la sortie et la valeur 1 ou 0 qu'elle peut prendre selon la ligne du tableau qui la décrit.

On notera : $v (F = 0) = \bar{F}$

$v (F = 1) = F$

2.5. Fonction véracité d'une ligne du tableau.

On définit la fonction véracité d'une ligne comme le produit des fonctions véracité des entrées et de la sortie.

Dans notre exemple :

Ligne 1 : $v_1 = \bar{a}_1 \bar{a}_2 a_3 F$

Ligne 2 : $v_2 = a_1 \bar{a}_2 \bar{a}_3 \bar{F}$

Ligne 3 : $v_3 = a_1 \bar{a}_2 a_3 F$

Ligne 4 : $v_4 = a_1 a_2 a_3 \bar{F}$

2.6. Fonction véracité du tableau entier.

Le fait que le système doit se trouver dans un des états qui le décrivent s'exprime par la somme logique des fonctions véracité de chaque ligne :

$$V = v_1 + v_2 + v_3 + v_4$$

Notons par A_i tout état d'entrée pour lequel F = 1

Notons par B_i tout état d'entrée pour lequel F = 0

On peut écrire : $$v = (\sum_{i \in I_1} A_i)F + (\sum_{i \in I_2} B_i)\bar{F}$$

De même en posant : $$\sum_{i \in I_1} A_i = A$$

et : $$\sum_{i \in I_2} B_i = B$$

on obtient : $$V = AF + B\bar{F}$$

V est la fonction véracité du tableau entier. C'est la fonction booléenne décrivant le système équivalente au tableau de correspondance. Elle n'est fonction que des états définis par le tableau !

Cette fonction prend la valeur 1 lorsque le système se trouve dans un état correspondant à une ligne du tableau.

2.7. Introduction des états indéfinis dans la fonction véracité du système.

Montrons qu'on peut introduire implicitement les états indéfinis par le raisonnement suivant :

Il suffit d'exprimer le fait que chaque état d'entrée du tableau "implique" l'état de sortie correspondant. C'est-à-dire que chaque A_i implique F et chaque B_i implique $\bar{F}$.

$$A_i \Longrightarrow F \qquad \forall i \in I_1$$

$$B_i \Longrightarrow \bar{F} \qquad \forall i \in I_2$$

Pour le montrer introduisons une nouvelle fonction véracité W caractéristique de l'ensemble des relations

d'implication entre les entrées et la sortie du tableau :

$$W = \prod_{i \in I_1} (F + \bar{A}_i) \prod_{i \in I_2} (\bar{F} + \bar{B}_i)$$

En développant :

$$(F + \bar{A}_1)(F + \bar{A}_2)(\dots) \dots = F + \bar{A}_1\bar{A}_2 \dots = F + \bar{A}$$

car : $A = A_1 + A_2 + \dots \iff \bar{A} = \bar{A}_1\bar{A}_2 \dots$

par De Morgan.

De même : $(\bar{F} + \bar{B}_1)(\bar{F} + \bar{B}_2)(\dots) = \bar{F} + \bar{B}$

Finalement : $W = (F + \bar{A})(\bar{F} + \bar{B})$

Montrons sous quelles conditions cette nouvelle fonction décrit le système :

Pour ce faire, nous introduisons une nouvelle fonction véracité R, caractéristique de l'égalité de V = W.

On pose :
$$\begin{aligned} R(V=W) &= VW + \bar{V}\bar{W} \\ &= (AF + B\bar{F})(F + \bar{A})(\bar{F} + \bar{B}) \\ &\quad + (\bar{A} + \bar{F})(\bar{B} + F)(\bar{F}A + FB) \\ &= (A\bar{B} + \bar{A}B)F + (A\bar{B} + \bar{A}B)\bar{F} \\ &= A\bar{B} + \bar{A}B \end{aligned}$$

Or : $AB = 0$ car un état des entrées ne peut pas impliquer simultanément F et $\bar{F}$.

D'où : $R = A\bar{B} + \bar{A}B + AB = A + B = \overline{(\bar{A}\bar{B})}$

On a donc le résultat suivant :

$$R(V = W) = A + B$$

Ainsi les 2 fonctions sont égales lorsque A + B = 1, ce qui est le cas pour tous les états d'entrée définis du système.

En fait, W = V + $\bar{A}\bar{B}$ où $\bar{A}\bar{B}$ représente la fonction qui contient tous les états indéfinis du système.

En effet : $W = (F + \bar{A})(\bar{F} + \bar{B}) = \bar{B}F + \bar{A}\bar{F} + \bar{A}\bar{B}$

$= \bar{B}AF + \bar{B}\bar{A}F + \bar{A}B\bar{F} + \bar{A}\bar{B}\bar{F} + \bar{A}\bar{B}$

$= AB\bar{F} + \bar{A}B\bar{F} + \bar{A}\bar{B}$

Et si AB = 0 :

$W = AB\bar{F} + ABF + \bar{A}B\bar{F} + AB\bar{F} + \bar{A}\bar{B}$

$W = AF + B\bar{F} + \bar{A}\bar{B} = V + \bar{A}\bar{B}$!

Il n'est pas nécessaire d'introduire explicitement les états indéfinis dans V pour obtenir W :

On part de : $V = AF + B\bar{F}$

En inversant : $\bar{V} = (\bar{A} + \bar{F})(\bar{B} + F) = \bar{A}F + \bar{B}\bar{F} + \bar{A}\bar{B}$

Il suffit alors d'inverser chaque variable de sortie pour obtenir $W = \bar{B}F + \bar{A}\bar{F} + \bar{A}\bar{B}$!

D'où le théorème :

Théorème 4 : La fonction véracité du système avec états indéfinis s'obtient par inversion de la fonction véracité du système avec états définis après inversion des variables de sortie.

On part de : $V = AF + B\bar{F}$

On effectue l'inversion des variables de sortie pour obtenir une nouvelle fonction Y :

$$Y = A\bar{F} + BF$$

Et on peut écrire : $W = \bar{Y}$

L'inversion de $\bar{Y}$ fournit tous les impliquants premiers de W. Ceux-ci s'interprètent de manière analogue aux impliquants de V. C'est-à-dire qu'à un impliquant premier CF de W correspond un impliquant premier C de F en tenant compte des états indéfinis.

Même raisonnement pour un terme $C\bar{F}$.

Un impliquant premier C de W peut être considéré comme impliquant de F ou de $\bar{F}$, c'est donc un impliquant premier ne contenant que des états indéfinis.

Nous trouvons une démonstration de ce raisonnement au chap. 4.5. théorème 7.

Exemple du § 2.2. :

La fonction véracité du tableau s'écrit :

$$V = \bar{a}_1\bar{a}_2a_3F + a_1\bar{a}_2\bar{a}_3\bar{F} + a_1\bar{a}_2a_3F + a_1a_2a_3\bar{F}$$

Inversons les variables de sortie :

$$Y = \bar{a}_1\bar{a}_2a_3\bar{F} + a_1\bar{a}_2\bar{a}_3F + a_1\bar{a}_2a_3\bar{F} + a_1a_2a_3F$$

En calculant $W = \bar{Y}$:

$$P_0 = 1$$

$$P_1 = P_0(a_1 + a_2 + \bar{a}_3 + F) = a_1 + a_2 + \bar{a}_3 + F$$

$$P_2 = P_1(\bar{a}_1 + a_2 + a_3 + \bar{F}) = a_1a_3 + a_1\bar{F} + a_2 + \bar{a}_1\bar{a}_3 + \bar{a}_3\bar{F} + \bar{a}_1F + a_3F$$

$$P_3 = P_2(\bar{a}_1 + a_2 + \bar{a}_3 + F) = \cancel{a_1a_3F} + \cancel{a_1\bar{a}_3\bar{F}} + a_2 + \bar{a}_1\bar{a}_3 + \bar{a}_3\bar{F} + \bar{a}_1F + a_3F$$

$$P_4 = P_3(\bar{a}_1 + \bar{a}_2 + \bar{a}_3 + \bar{F}) = \bar{a}_1a_2 + a_2\bar{a}_3 + a_2\bar{F} + \bar{a}_1\bar{a}_3 + \bar{a}_3\bar{F} + \bar{a}_1F + \cancel{\bar{a}_1a_3F} + \bar{a}_2a_3F$$

$$W = F(\bar{a}_1 + \bar{a}_2a_3) + \bar{F}(a_2 + \bar{a}_3) + \bar{a}_1a_2 + a_2\bar{a}_3 + \bar{a}_1\bar{a}_3$$

On a donc obtenu en une fois les impliquants premiers de F :

$$\bar{a}_1 \ , \ \bar{a}_2a_3$$

et les impliquants premiers de $\bar{F}$:

$$a_2 \ , \ \bar{a}_3$$

ainsi que les impliquants premiers ne contenant que des états indéfinis :

$$\bar{a}_1a_2 \ , \ a_2\bar{a}_3 \ , \ \bar{a}_1\bar{a}_3$$

2.8. Borne inférieure, borne supérieure d'une fonction incomplètement définie.

On a vu que l'équation booléenne qui définit F est :

$$W = (F + \bar{A})(\bar{F} + \bar{B}) = 1$$

On peut résoudre cette équation par rapport à F ;

Il faut imposer :

$$F + \bar{A} = 1 \quad \text{et} \quad \bar{F} + \bar{B} = 1$$

Ce qui donne : $F \geqslant A$

$F \leqslant \bar{B}$

Donc tout F tel que :

$$A \leqslant F \leqslant \bar{B}$$

est solution de W = 1 !

On appelle conventionnellement :

A : borne inférieure de F

B : borne supérieure de F

2.9. Calcul des impliquants premiers de la borne supérieure.

Souvent dans la pratique, on ne désire calculer que les impliquants premiers de la borne supérieure de F. Il s'agit donc de s'éviter des calculs inutiles.

Nous avons vu au chap. 2.7. :

$$W = \bar{B}F + \bar{A}\bar{F} + \bar{A}\bar{B}$$

On veut obtenir $\bar{B}F$ par le calcul, en supprimant $\bar{A}\bar{F}$ de l'expression, qui représente la borne supérieure de $\bar{F}$ jugée inutile.

Pour ce faire, on définit la fonction véracité de la borne supérieure de F, notée $[W-]$, qu'on obtient en "masquant" le terme $\bar{A}\bar{F}$ présent dans W par l'addition du terme $\bar{F}$:

$$[W-] = W + \bar{F} = \bar{F} + \bar{A}\bar{F} + \bar{B}F + \bar{A}\bar{B}$$

Après un bref calcul, on trouve :

$$[W-] = \bar{F} + \bar{B}$$

Or $[W-]$ s'obtient par inversion de :

$$[Y-] = FB$$

défini de manière analogue à Y (chap. 2.7. page) :

$[Y-]$ s'obtient par inversion de la variable de sortie de :

$$[V-] = B\bar{F}$$

où $[V-]$ s'obtient à partir de $V = AF + B\bar{F}$ en multipliant l'expression par $\bar{F}$.

D'où la règle simple :

Pour obtenir les impliquants premiers de la borne supérieure du système à partir de sa fonction véracité, il suffit de multiplier celle-ci par $\bar{F}$, ce qui revient simplement à ignorer les termes comprenant F !

Exemple du § 2.2. :

$$V = (\bar{a}_1\bar{a}_2a_3 + a_1\bar{a}_2a_3)F + (a_1\bar{a}_2\bar{a}_3 + a_1a_2a_3)\bar{F}$$

$$[V\text{-}] = V\bar{F} = (a_1\bar{a}_2\bar{a}_3 + a_1a_2a_3)\bar{F}$$

$$[Y\text{-}] = (a_1\bar{a}_2\bar{a}_3 + a_1a_2a_3)F$$

d'où :

$$[W\text{-}] = [\bar{Y}\text{-}] = \bar{F} + (\bar{a}_1 + a_2 + a_3)(\bar{a}_1 + \bar{a}_2 + \bar{a}_3)$$
$$= \bar{a}_1 + a_2\bar{a}_3 + \bar{a}_2a_3 + \bar{F}$$

Les impliquants de la borne supérieure de F sont donc :

$$\bar{a}_1 + a_2\bar{a}_3 + \bar{a}_2a_3$$

Le terme $\bar{F}$ est à ignorer; il a en effet été introduit pour le masquage.

On constate d'autre part qu'il n'est plus possible de distinguer les impliquants premiers ne contenant que des états indéfinis. (cf. chap. 2.8., page 34).

CHAPITRE 3. CALCUL DES SOMMES IRREDONDANTES A L'AIDE D'UN ALGORITHME DE CALCUL DES IMPLIQUANTS PREMIERS.

3.1. Introduction.

Nous avons vu au chap. 1.5. que toute somme irrédondante n'est composée que d'impliquants premiers. L'algorithme du chap. 1.6. permet de les calculer tous. Les termes de chaque somme irrédondante forment un sous-ensemble de l'ensemble des impliquants premiers.

La méthode présentée ci-après consiste à caractériser chaque impliquant premier par une variable auxiliaire d'indice. Ceci permet de ramener le problème de la réduction du nombre de termes de la somme au calcul des impliquants premiers d'une fonction caractéristique où apparaissent ces variables auxiliaires.

3.2. Démonstration de la méthode.

On dispose de l'ensemble des G_i $i \in I$ des impliquants premiers de la fonction F : satisfaisant la relation :

$$A \leq \sum_{i \in I} G_i$$

ou A est la borne inférieure de F.

On cherche un sous-ensemble G_S avec $S \subseteq I$ tel que :

$$G_S = \sum_{i \in S} G_i \qquad \text{vérifie} \quad A \leq G_S$$

Pour ce faire, on forme la fonction :

$$G(X) = \sum_{i \in I} G_i X_i$$

où les X_i sont des variables de choix indépendantes.

Montrons qu'à tout $G_S \nsubseteq A$ correspond et réciproquement un impliquant :

$$X_S = \prod_{i \in S} X_i$$

appartenant à la fonction véracité :

$$V(X) = V(A \leq \sum_{i \in I} G_i X_i)$$

$$V(X) = \bar{A} + G(X)$$

au moyen du théorème suivant :

Théorème 5 :

$$A \leq G_S \quad \Longleftrightarrow \quad X_S < V(X)$$

Démonstration :

a) La fonction véracité de l'implication ($\Longrightarrow$) s'écrit :

$$A \prod_{i \in S} \bar{G}_i + \sum_{i \in S} \bar{X}_i + \bar{A} + \sum_{i \in I} G_i X_i =$$

$$\prod_{i \in S} \bar{G}_i + \sum_{i \in S} (\bar{X}_i + G_i) + \ldots =$$

$$\overline{\sum_{i \in S} G_i} + \sum_{i \in S} G_i + \ldots = 1$$

b) Supposons :

$$\prod_{i \in S} X_i \leq \bar{A} + \sum_{i \in I} G_i X_i$$

vérifié. La relation est donc vraie pour toute valeur des variables X_i, donc aussi dans le cas particulier :

$$X_i = 1 \qquad \forall \; i \in S$$

$$X_i = 0 \qquad \forall \; i \in I\text{-}S$$

d'où :

$$1 = \bar{A} + \sum_{i \in S} G_i \qquad \text{cqfd.}$$

Commentaire :

1) Tout impliquant de V(X) de la forme $X_S = \prod_{i \in S} X_i$, c'est-à-dire ne dépendant que des variables auxiliaires X_i, fournissant une solution $G_S \geqslant A$, il reste à montrer que tout impliquant premier de cette forme fournit une couverture irrédondante de A. (La démonstration est évidente).

2) L'intérêt de cette méthode réside dans le fait que le même algorithme permet de trouver successivement les impliquants premiers d'une fonction et ses couvertures premières.

3.3. Exemple :

Soit : $F = ab + \bar{b}c$

dont on a calculé les impliquants premiers (chap. 1.6.2) :

$$G_I = G_1 + G_2 + G_3 = ab + \bar{b}c + ac \qquad i = 1,2,3$$

Il s'agit de satisfaire la relation :

$$A \leqslant X_1G_1 + X_2G_2 + X_3G_3$$

par le calcul des impliquants premiers X_S de :

$$V(X) = \bar{A} + X_1G_1 + X_2G_2 + X_3G_3$$

$$V(X) = \bar{A} + G(X)$$

Calculons : $V(X) = \bar{A} + G(X)$

en inversant la relation :

$$\bar{V}(X) = A \cdot \bar{G}(X) = (ab + \bar{b}c)\overline{(X_1ab + X_2\bar{b}c + X_3ac)}$$

On procède de manière itérative, selon l'algorithme du chap. 1.6.2. (remarque 3)

$$P_0 = ab + \bar{b}c$$

$$P_1 = P_0(\bar{x}_1 + \bar{a} + \bar{b}) = ab\bar{x}_1 + \bar{b}c$$

$$P_2 = P_1(\bar{x}_2 + b + \bar{c}) = ab\bar{x}_1 + \bar{b}c\bar{x}_2$$

$$\bar{V}(x) = P_3 = P_2(\bar{x}_3 + \bar{a} + \bar{c}) = ab\bar{x}_1\bar{x}_3 + abc\bar{\bar{x}}_1 + \bar{b}c\bar{x}_2\bar{x}_3 + \bar{\bar{a}}\bar{b}c\bar{x}_2$$

puis on calcule V(x) :

$$P_0 = 1$$

$$P_1 = P_0(\bar{a} + \bar{b} + x_1 + x_3) = \bar{a} + \bar{b} + x_1 + x_3$$

$$P_2 = P_1(\bar{a} + \bar{b} + c + x_1) = \bar{a} + \bar{b} + x_1 + cx_3 + x_1x_3$$

$$P_3 = P_2(b + \bar{c} + x_2 + x_3) = \bar{a}b + \bar{a}\bar{c} + \bar{a}x_2 + \bar{a}x_3 + \bar{b}\bar{c} + \bar{b}x_2 + \bar{b}x_3 + bx_1 + \bar{c}x_1 + x_1x_2 + x_1x_3 + bcx_3 + cx_3$$

$$V(x) = P_4 = P_3(a + b + \bar{c} + x_2) = \bar{a}b + \bar{a}\bar{c} + \bar{a}x_2 + \bar{b}\bar{c} + \bar{b}x_2 + a\bar{b}x_3 + bx_1 + \bar{c}x_1 + x_1x_2 + ax_1x_3 + acx_3 + bcx_3 + cx_2x_3$$

Le seul impliquant X_S est x_1x_2, qui fournit la somme irrédondante :

$$G_S = ab + \bar{b}c$$

Le calcul des X_S paraît visiblement lourd. Il peut cependant être allégé en tenant compte de la considération suivante :

Tout terme contenant une variable autre que $x_i, i \notin I$, apparaissant au cours de l'itération ne peut donner un terme X_S. On effectue donc le calcul en ne tenant compte que des x_i.

Dans notre exemple, l'inversion de $\bar{V}(X)$ se réduit alors à :

$$V(X) = (x_1 + x_3)\ x_1(x_2 + x_3)\ x_2 = x_1x_2$$

ce qui représente un calcul visiblement plus court !

3.4. Interprétation de la fonction $\bar{V}(X)$.

On a vu au chap. 3.2. que V(X) est la fonction véracité de :

$$A \leq G(X)$$

Or si A est donné sous forme de somme de produits :

$$A = \sum_{j \in J} A_j$$

alors V(X) est le produit des fonctions véracité des relations partielles :

$$A_j \leq G(X) \qquad \forall\ j \in J$$

c'est-à-dire $V_j(X) = \bar{A}_j + G(X)$

en effet : $V(X) = \bar{A} + G(X) = \prod_{j \in J} \bar{A}_j + G(X) = \prod_{j \in J} V_j(X)$

d'où on tire $\bar{V}(X) = \sum_{j \in J} \bar{V}_j(X)$

L'interprétation des impliquants de $\bar{V}(X)$ peut se faire grâce au théorème suivant :

Théorème 6 :

$$G_i \cdot A_j = 0 \quad \forall\ i \in I\text{-}T \iff A_j \prod_{i \in T} \bar{X}_i \leq \bar{V}_j(X) \leq \bar{V}(X)$$

Démonstration :

a) La fonction véracité de l'implication ($\Longrightarrow$) s'écrit :

$$A_j \sum_{i \in I-T} G_i + \bar{A}_j + \sum_{i \in T} X_i + A_j \prod_{i \in I} (\bar{G}_i + \bar{X}_i) =$$

$$\sum_{i \in I-T} G_i + \sum_{i \in T} X_i + \prod_{i \in I} (\bar{G}_i + \bar{X}_i) + \ldots \quad =$$

$$\sum_{i \in I-T} G_i + \sum_{i \in T} X_i + \prod_{i \in I-T} \bar{G}_i \prod_{i \in T} \bar{X}_i + \ldots \quad = 1$$

b) Supposons la relation :

$$A_j \prod_{i \in T} \bar{X}_i \leq A_j \prod_{i \in I} (\bar{G}_i + \bar{X}_i)$$

vérifiée. Elle est donc vraie pour toute valeur des variables indépendantes X_i, donc en particulier pour :

$$X_i = 0 \quad \forall\, i \in T$$

$$X_i = 1 \quad \forall\, i \in I-T$$

d'où : $A_j \leq A_j \prod_{i \in I-T} \bar{G}_i \Longrightarrow \bar{A}_j + A_j \prod_{i \in I-T} \bar{G}_i = 1$

Soit : $A_j \sum_{i \in I-T} G_i = 0$ cqfd.

Remarque : Pour des raisons de clarté, nous n'avons pas distingué les termes A_j donnés par : $F = \sum_{j \in J} A_j$ et les termes A_j qui apparaissent dans les impliquants de $\bar{V}(X)$. Cependant le théorème ci-dessus reste valable, car les seconds sont égaux aux premiers si ce sont des mintermes, et leur sont inclus dans le cas général.

3.4.2 Cas particulier.

Nous considérons dorénavant l'ensemble S tel que $G_i A_j = 0 \quad \forall\, i \in I\text{-}S$ et tel qu'il n'existe pas $S' \subset S$ avec $G_i A_j = 0 \quad \forall\, i \in I\text{-}S'$.

Alors, l'impliquant $A_j \prod_{i \in S} \bar{X}_i$ de $\bar{V}(X)$ est "premier en X", ce qui signifie qu'il n'existe pas un ensemble $S' \subset S$ tel que :

$$A_j \prod_{i \in S'} \bar{X}_i \leq \bar{V}(X)$$

3.4.3 Commentaires :

Chaque impliquant "premier en X" $A_j \prod_{i \in S} \bar{X}_i$ fournit l'ensemble de tous les impliquants G_i, $i \in I\text{-}S$ qui ont une intersection nulle avec A_j et par conséquent, chaque G_i, $i \in S$ a une intersection non nulle avec A_j.

L'expression $A_j \prod_{i \in S} \bar{X}_i \leq \bar{V}(X)$ signifie qu'il est nécessaire et suffisant d'exclure de $\sum G_i$, $i \in I$ tous les termes G_i, $i \in S$ pour que A_j ne soit pas couvert, même partiellement, c'est-à-dire pour que $\bar{V}(X) = 1$.

Autrement dit, chaque G_i, $i \in S$ a une intersection non nulle avec A_j et est donc susceptible de couvrir A_j, ne serait-ce que partiellement.

Dans le cas particulier ou A_j est un minterme, chaque G_i , $i \in S$, est une couverture de A_j. (évident)

Nous admettons sans démonstration que l'obtention de $\bar{V}(X)$ par l'algorithme de calcul des impliquants premiers du chap. 1.6.2.,

$$\bar{V}(X) = A \cdot \overline{(\sum_{i \in I} G_i X_i)}$$

ne fournit que des impliquants "premiers en X".

En conclusion, on peut considérer $\bar{V}(X)$ comme une représentation algébrique booléenne d'un "tableau de couverture" selon Petrick. Illustrons ce fait par l'exemple ci-dessous.

3.5. Calcul d'une fonction simple en sommes irrédondantes.

Soit la fonction définie par le tableau :

Exemple du § 1.6.3.

a	b	c	d	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

La fonction véracité de ce tableau s'écrit :

$$V = \bar{a}\bar{b}\bar{c}\bar{d}\bar{F} + \bar{a}\bar{b}\bar{c}dF + \bar{a}\bar{b}c\bar{d}F + \bar{a}\bar{b}cdF + \bar{a}b\bar{c}\bar{d}\bar{F} + \bar{a}b\bar{c}dF$$
$$+ \bar{a}bc\bar{d}F + \bar{a}bcdF + a\bar{b}\bar{c}\bar{d}F + a\bar{b}\bar{c}d\bar{F} + a\bar{b}c\bar{d}\bar{F} + a\bar{b}cd\bar{F}$$
$$+ ab\bar{c}\bar{d}F + ab\bar{c}dF + abc\bar{d}\bar{F} + abcd\bar{F}$$

Le calcul des impliquants premiers de F, se calcule d'après la relation du § 2.9. par :

$$[W-] = \bar{F} + \bar{B}$$

C'est-à-dire en inversant V, après avoir inversé chaque variable de sortie, et en ignorant les termes contenant la variable de sortie F.

Dans notre cas :

$$[W-] = \bar{F} + \overline{(\bar{a}b\bar{c}\bar{d} + \bar{a}b\bar{c}d + \bar{a}bc\bar{d} + ab\bar{c}d + abc\bar{d} + abcd + abcd)}$$

$P_0 = 1$

$P_1 = a + b + c + d$

$P_2 = P_1(a + b + c + \bar{d}) = a + b + c$

$P_3 = P_2(a + \bar{b} + c + d) = a + c + bd$

$P_4 = P_3(\bar{a} + b + c + \bar{d}) = ab + a\bar{d} + c + bd$

$P_5 = P_4(\bar{a} + b + \bar{c} + d) = ab + ac\bar{d} + \bar{a}c + bc + cd + bd$

$P_6 = P_5(\bar{a} + b + \bar{c} + \bar{d}) = ab + ac\bar{d} + \bar{a}c + bc + bd$

$P_7 = P_6(\bar{a} + \bar{b} + \bar{c} + d) = ab\bar{c} + \cancel{abd} + ac\bar{d} + \bar{a}c + \cancel{\bar{a}bc} + \cancel{bcd} + bd$

$P_8 = P_7(\bar{a} + \bar{b} + \bar{c} + \bar{d}) = ab\bar{c} + ac\bar{d} + \bar{a}c + \bar{a}bd + b\bar{c}d$

d'où :

$$[W-] = \bar{F} + ab\bar{c} + ac\bar{d} + \bar{a}c + \bar{a}bd + b\bar{c}d$$

La borne supérieure de F contient donc cinq impliquants premiers. Pour trouver les sommes irrédondantes, il faut donc chercher les valeurs des x_i, $i = 1,2,3,4,5$ telles que :

$$A \leqslant ab\bar{c}x_1 + a\bar{c}\bar{d}x_2 + \bar{a}cx_3 + \bar{a}bdx_4 + b\bar{c}dx_5$$

c'est-à-dire calculer tout d'abord :

$$\bar{V}(x) = \overline{A \cdot (ab\bar{c}x_1 + a\bar{c}\bar{d}x_2 + \bar{a}cx_3 + \bar{a}bdx_4 + b\bar{c}dx_5)}$$

Procédons selon l'algorithme d'inversion :

$$P_0 = A = \bar{a}\bar{b}c\bar{d} + \bar{a}\bar{b}cd + \bar{a}b\bar{c}d + \bar{a}bc\bar{d} + \bar{a}bcd + a\bar{b}\bar{c}\bar{d} + ab\bar{c}\bar{d} + ab\bar{c}d$$

$$P_1 = P_0(\bar{a} + \bar{b} + c + \bar{x}_1) = \bar{a}\bar{b}c\bar{d} + \bar{a}\bar{b}cd + \bar{a}b\bar{c}d + \bar{a}bc\bar{d} + \bar{a}bcd + a\bar{b}\bar{c}\bar{d} + ab\bar{c}\bar{d}\bar{x}_1$$

$$P_2 = P_1(\bar{a} + b + c + \bar{x}_2) = \bar{a}\bar{b}c\bar{d} + \bar{a}\bar{b}cd + \bar{a}b\bar{c}d + \bar{a}bc\bar{d} + \bar{a}bcd + a\bar{b}\bar{c}\bar{d}\bar{x}_2 + ab\bar{c}\bar{d}\bar{x}_1\bar{x}_2 + ab\bar{c}d\bar{x}_1$$

$$P_3 = P_2(a + \bar{c} + \bar{x}_3) = \bar{a}\bar{b}c\bar{d}\bar{x}_3 + \bar{a}\bar{b}cd\bar{x}_3 + \bar{a}b\bar{c}d + \bar{a}bc\bar{d}\bar{x}_3 + \bar{a}bcd\bar{x}_3 + a\bar{b}\bar{c}\bar{d}\bar{x}_2 + ab\bar{c}\bar{d}\bar{x}_1\bar{x}_2 + ab\bar{c}d\bar{x}_1$$

$$P_4 = P_3(a + \bar{b} + \bar{d} + \bar{x}_4) = \bar{a}\bar{b}c\bar{d}\bar{x}_3 + \bar{a}\bar{b}cd\bar{x}_3 + \bar{a}b\bar{c}d\bar{x}_4 + \bar{a}bc\bar{d}\bar{x}_3 + \bar{a}bcd\bar{x}_3\bar{x}_4 + a\bar{b}\bar{c}\bar{d}\bar{x}_2 + ab\bar{c}\bar{d}\bar{x}_1\bar{x}_2 + ab\bar{c}d\bar{x}_1$$

$$\bar{V}(x) = P_5 = P_4(\bar{b} + c + \bar{d} + \bar{x}_5) = \bar{a}\bar{b}c\bar{d}\bar{x}_3 + \bar{a}\bar{b}cd\bar{x}_3 + \bar{a}b\bar{c}d\bar{x}_4\bar{x}_5 + \bar{a}bc\bar{d}\bar{x}_3 + \bar{a}bcd\bar{x}_3\bar{x}_4 + a\bar{b}\bar{c}\bar{d}\bar{x}_2 + ab\bar{c}\bar{d}\bar{x}_1\bar{x}_2 + ab\bar{c}d\bar{x}_1\bar{x}_5$$

L'interprétation de $\bar{V}(x)$ d'après le chap. 3.4.1. peut se faire comme suit :

L'impliquant $\bar{a}\bar{b}c\bar{d}$ de F a une intersection non nulle avec l'impliquant premier $\bar{a}c$ indicé par x_3.

De même pour $\bar{a}\bar{b}cd$ et $\bar{a}bc\bar{d}$.

L'impliquant $\bar{a}bc\bar{d}$ a une intersection non nulle avec les impliquants premiers $\bar{a}bd$ indicé par x_4 et $bc\bar{d}$ indicé par x_5.

etc...

De plus, le fait qu'un impliquant de F ayant une intersection non nulle avec un impliquant premier, soit un minterme, entraîne qu'il est couvert par cet impliquant premier.

Dans notre exemple, $\bar{V}(x)$ fournit les relations suivantes :

$$\bar{a}\bar{b}c\bar{d} \leqslant \bar{a}c$$

$$\bar{a}\bar{b}cd \leqslant \bar{a}c$$

$$\bar{a}bc\bar{d} \leqslant \bar{a}bd \text{ et } \bar{a}bc\bar{d} \leqslant bc\bar{d}$$

etc...

La fonction $\bar{V}(x)$ est l'équivalent algébrique d'un tableau de couverture de Petrick .

Les solutions de V(x) ne dépendant que des variables x_i s'obtiennent par inversion de $\bar{V}(x)$, en ne gardant que les x_i :

$$V(x) = x_3(x_4 + x_5)(x_3 + x_4)x_2(x_1 + x_2)(x_1 + x_5)$$

$$V(x) = x_2x_3x_5 + x_1x_2x_3x_4$$

Ce qui donne les 2 sommes irrédondantes :

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

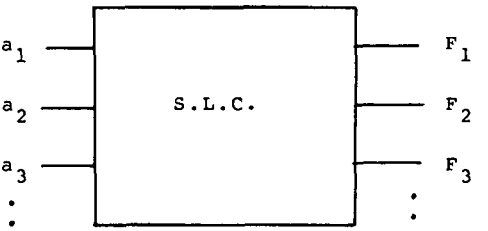
$$F = ab\bar{c} + a\bar{c}\bar{d} + \bar{a}c + \bar{a}bd$$

CHAPITRE 4. CALCUL DES IMPLIQUANTS PREMIERS D'UN SYSTEME DE FONCTIONS.

Nous avons vu au chapitre 2 comment calculer les impliquants premiers d'une fonction booléenne avec états indéfinis.

Généralisons la méthode aux systèmes de plusieurs fonctions, de manière à pouvoir utiliser l'algorithme d'inversion d'une fonction booléenne unique.

4.1. Définition du système.



On peut décrire le comportement de ce système de manière analogue à un système à sortie unique :

Par exemple :

a_1	a_2	a_3	F_1	F_2
0	0	1	1	1
1	0	0	0	0
1	0	1	1	0
1	1	1	ϕ	1

Décrivons ce système au moyen d'une fonction booléenne unique :

4.2. Fonction véracité du système.

On la définit de manière analogue au système à sortie unique, par la somme des fonctions véracité de chaque ligne :

$$V = \sum_{i \in I} E_i S_i = \sum_{i \in I} E_i(a_1, a_2, \ldots, a_N)\ S_i\ (F_1, F_2, \ldots, F_J)$$

avec : I = nombre de lignes définies par la table

N = nombre de variables d'entrée

J = nombre de variables de sortie.

et on appelle :

E_i : état d'entrée de la ligne i

S_i : état de sortie de la ligne i.

Pour l'exemple du chap. 4.1., on a :

$$V = \bar{a}_1\bar{a}_2a_3F_1F_2 + a_1\bar{a}_2\bar{a}_3\bar{F}_1\bar{F}_2 + a_1\bar{a}_2a_3F_1\bar{F}_2 + a_1a_2a_3F_2$$

Remarques :

a) Cette fonction véracité vaut 1 si le système se trouve dans un état défini et 0 dans le cas contraire.

b) Cet exemple contient 2 types d'indéfinitions. D'une part, les états d'entrée qui n'apparaissent pas dans le tableau et par conséquent pour lesquels les états de sortie ne sont pas spécifiés,

d'autre part les états d'entrée définis dans le tableau pour lesquels l'état des sorties n'est pas complètement défini.

Dans notre exemple, les états d'entrée :

$\bar{a}_1\bar{a}_2\bar{a}_2$, $\bar{a}_1 a_2\bar{a}_3$, $\bar{a}_1 a_2 a_3$, $a_1\bar{a}_2\bar{a}_3$

sont du premier type, alors que l'état :

$a_1 a_2 a_3$ F_2 n'est pas défini pour F_1.

On peut alors interpréter cette ligne comme étant la forme contractée de :

$a_1 a_2 a_3 (F_1 + \bar{F}_1)$ F_2.

Il est donc important de constater que la fonction véracité V contient les états indéfinis du second type mentionné ci-dessus.

Quant aux états indéfinis du premier type, on les introduit implicitement de la même manière que pour un système à sortie unique.

4.3. Fonction véracité du système avec états indéfinis du premier type.

Montrons que pour obtenir la fonction véracité du système avec les états indéfinis du premier type, il faut exprimer le fait que l'état d'entrée <u>implique</u> l'état de sortie correspondant et ceci pour chaque ligne définie par le tableau. Cette fonction s'écrit :

$$W = \prod_{i \in I} W_i \ (E_i \Longrightarrow S_i)$$

c'est-à-dire : $$W = \prod_{i \in I} (\bar{E}_i + S_i)$$

Il s'agit de montrer que :

$$W = V + \sum_{i \notin I} E_i$$

où : $\sum_{i \notin I} E_i$ représente la somme des états d'entrée qui ne figurent pas dans le tableau.

On a les relations évidentes :

$$\sum_{i \notin I} E_i + \sum_{i \in I} E_i = 1$$

$$\sum_{i \notin I} E_i \cdot \sum_{i \in I} E_i = 0$$

d'où on tire : $\sum_{i \notin I} E_i = \overline{\sum_{i \in I} E_i} = \prod_{i \in I} \bar{E}_i$

Nous devons donc montrer que :

$$W = V + \prod_{i \in I} \bar{E}_i$$

Pour simplifier la démonstration, montrons successivement que :

$$W \leq V + \prod_{i \in I} \bar{E}_i$$

et $$W \not\geq V + \prod_{i \in I} \bar{E}_i$$

a) La fonction véracité de la relation $\leq$ s'exprime :

$$R\left(W \leq V + \prod_{i \in I} \bar{E}_i\right) = \bar{W} + V + \prod_{i \in I} \bar{E}_i$$

et en introduisant : $W = \prod_{i \in I} (\bar{E}_i + S_i)$

$$V = \sum_{i \in I} E_i S_i$$

on obtient : $R = \sum_{i \in I} E_i \bar{S}_i + \sum_{i \in I} E_i S_i + \prod_{i \in I} \bar{E}_i = 1$

b) De manière analogue :

$$R(W \gg V + \prod_{i \in I} \bar{E}_i) = W + \bar{V} \cdot \sum_{i \in I} E_i$$

$$= \prod_{i \in I} (\bar{E}_i + S_i) + \prod_{i \in I} (\bar{E}_i + \bar{S}_i) \sum_{i \in I} E_i$$

d'où : $\bar{R} = (\sum_{i \in I} E_i \bar{S}_i)(\sum_{i \in I} E_i S_i + \prod_{i \in I} \bar{E}_i)$

en développant l'expression, on trouve :

$$\bar{R} = (E_1 \bar{S}_1 + E_2 \bar{S}_2 + E_3 \bar{S}_3 + \ldots)(E_1 S_1 + E_2 S_2 + E_3 S_3 + \ldots)$$

$$\bar{R} = E_1 E_2 (\bar{S}_1 S_2 + S_1 \bar{S}_2) + E_1 E_3 (\bar{S}_1 S_3 + S_1 \bar{S}_3) + \ldots$$

$$\bar{R} = \sum_{i,j \in I} E_i E_j (\bar{S}_i S_j + S_i \bar{S}_j) = 0$$

car : $\forall i,j$ tq $S_i \neq S_j$, on a $E_i E_j = 0$

$\forall i,j$ tq $E_i E_j \neq 0$, on a $S_i = S_j$

En effet, les deux expressions ci-dessus représentent les conditions que l'on admet implicitement dans l'écriture du tableau. On peut les interpréter comme suit : pour deux états d'entrée définis dans la table et d'intersection non nulle, on ne peut spécifier pour toute fonction deux états différents.

•

Par la relation : $W = \prod_{i \in I} (\bar{E}_i + S_i)$

on déduit que : $\bar{W} = \sum_{i \in I} E_i \bar{S}_i$

or : $V = \sum_{i \in I} E_i S_i$

On retrouve le même procédé de calcul du paragraphe 2.7. (théorème 4) généralisé au cas de plusieurs fonctions. C'est-à-dire que pour obtenir les impliquants premiers de W, il suffit d'appliquer le processus d'inversion d'une fonction booléenne unique à la fonction V, dont chaque S_i est remplacé par $\bar{S}_i$.

4.4. Application à l'exemple du § 4.1.

On avait : $V = \bar{a}_1\bar{a}_2a_3F_1F_2 + a_1\bar{a}_2\bar{a}_3\bar{F}_1\bar{F}_2 + a_1\bar{a}_2a_3F_1\bar{F}_2 + a_1a_2a_3F_2$

d'où : $\bar{W} = \bar{a}_1\bar{a}_2a_3(\overline{F_1F_2}) + a_1\bar{a}_2\bar{a}_3(\overline{\bar{F}_1\bar{F}_2}) + a_1\bar{a}_2a_3(\overline{F_1\bar{F}_2}) + a_1a_2a_3\bar{F}_2$

Pour appliquer le processus d'inversion, il est nécessaire de développer les expressions de sortie :

$$\bar{W} = \bar{a}_1\bar{a}_2a_3\bar{F}_1 + \bar{a}_1\bar{a}_2a_3\bar{F}_2 + a_1\bar{a}_2\bar{a}_3F_1 + a_1\bar{a}_2\bar{a}_3F_2 + a_1\bar{a}_2a_3\bar{F}_1 + a_1\bar{a}_2a_3F_2 + a_1a_2a_3\bar{F}_2$$

On procède par étapes :

$$P_0 = 1$$

$$P_1 = P_0(a_1 + a_2 + \bar{a}_3 + F_1) = a_1 + a_2 + \bar{a}_3 + F_1$$

$$P_2 = P_1(a_1 + a_2 + \bar{a}_3 + F_2) = a_1 + a_2 + \bar{a}_3 + F_1F_2$$

$$P_3 = P_2(\bar{a}_1 + a_2 + a_3 + \bar{F}_1) = a_2 + \bar{a}_1\bar{a}_3 + \bar{a}_1F_1F_2 + a_1a_3 + a_3F_1F_2 + a_1\bar{F}_1 + \bar{a}_3\bar{F}_1$$

$$P_4 = P_3(\bar{a}_1 + a_2 + a_3 + \bar{F}_2) = a_2 + \bar{a}_1\bar{a}_3 + \bar{a}_1F_1F_2 + a_1a_3 + a_3F_1F_2 + a_1\bar{F}_1\bar{F}_2 + \bar{a}_3\bar{F}_1\bar{F}_2$$

$$P_5 = P_4(\bar{a}_1 + a_2 + \bar{a}_3 + F_1) = a_2 + \bar{a}_1\bar{a}_3 + \bar{a}_1F_1F_2 + \bar{a}_3\bar{F}_1\bar{F}_2 + a_1a_3F_1 + a_3F_1F_2$$

$$P_6 = P_5(\bar{a}_1 + a_2 + \bar{a}_3 + \bar{F}_2) = a_2 + \bar{a}_1\bar{a}_3 + \bar{a}_1F_1F_2 + \bar{a}_3\bar{F}_1\bar{F}_2 + a_1a_3F_1\bar{F}_2$$

$$P_7 = P_6(\bar{a}_1 + \bar{a}_2 + \bar{a}_3 + F_2) = \bar{a}_1a_2 + \bar{a}_1\bar{a}_3 + \bar{a}_1F_1F_2 + \bar{a}_3\bar{F}_1\bar{F}_2 + a_1\bar{a}_2a_3F_1\bar{F}_2 + a_2\bar{a}_3 + a_2F_2$$

$$W = \bar{a}_1a_2 + \bar{a}_1\bar{a}_3 + a_2\bar{a}_3 + \bar{a}_1F_1F_2 + \bar{a}_3\bar{F}_1\bar{F}_2 + a_1\bar{a}_2a_3F_1\bar{F}_2 + a_2F_2$$

Il est aisé de contrôler que :

$\bar{a}_1 a_2 + \bar{a}_1 a_3 + a_2 \bar{a}_3$ représente la somme des états indéfinis du premier type.

$\bar{a}_1 F_1 F_2$ signifie que $\bar{a}_1$ est impliquant premier du produit des fonctions F_1 et F_2. Etc...

Cependant, comme dans le cas de la synthèse d'une seule fonction à deux niveaux, on ne s'intéresse qu'aux impliquants des bornes supérieures de chaque fonction.

4.5. Calcul des impliquants premiers des bornes supérieures d'un système de fonctions.

La réalisation simultanée d'un système de plusieurs fonctions fait apparaître un critère d'optimisation spécifique, qui est le suivant :

un impliquant commun à plusieurs fonctions peut être utilisé pour la réalisation de plusieurs fonctions simultanément, ce qui aboutit à une diminution du coût total de la réalisation du système.

En effet, si : $A \leq F_1$ A premier

et : $B \leq F_2$ B premier

alors : $A \cdot B \leqslant F_1 \cdot F_2$

On voit que l'impliquant commun $A \cdot B$ n'est généralement pas impliquant premier de F_1 ni de F_2, mais est impliquant premier du produit F_1F_2.

Il s'agit donc de calculer les impliquants premiers des fonctions du système ainsi que ceux de tous leurs produits, en utilisant l'algorithme d'inversion d'une fonction booléenne unique.

Nous avons vu au chap. 2.9. que les impliquants premiers de la borne supérieure d'une fonction F_j incomplètement définies, s'obtiennent en calculant les impliquants premiers de la fonction véracité:

$$[W_j-] = \bar{F}_j + \bar{B}_j$$

où : $\bar{B}_j \geqslant F_j$ par définition de la borne supérieure.

Le calcul des impliquants premiers de $[W-] = \prod_{j \in J} [W_j-]$ fournit tous les impliquants premiers des produits des bornes supérieures des fonctions, en vertu du théorème suivant :

Théorème 7 : A tout impliquant de $[W-]$ de la forme $G \prod_{j \in J-S} \bar{F}_j$, G indépendant des variables F_j, correspond l'impliquant G du produit des bornes supérieures $\prod_{j \in S} \bar{B}_j$. $(S \subseteq J)$

I.e. :

$$G \leqslant \prod_{j \in S} \bar{B}_j \iff G \prod_{j \in J-S} \bar{F}_j \leqslant [W-]$$

Démonstration :

a) $G \leq \prod_{j \in S} \bar{B}_j \Rightarrow G \prod_{j \in J-S} \bar{F}_j \leq \prod_{j \in S} \bar{B}_j \prod_{j \in J-S} \bar{F}_j$

or : $\prod_{j \in S} \bar{B}_j \cdot \prod_{j \in J-S} \bar{F}_j \leq \prod_{j \in J} (\bar{B}_j + \bar{F}_j) = [W-]$ toujours

parce que le membre de gauche n'est que l'un des termes du membre de droite.

b) $G \prod_{j \in J-S} \bar{F}_j \leq [W-] \Leftrightarrow G \prod_{j \in J-S} \bar{F}_j \leq \prod_{j \in J} (\bar{F}_j + \bar{B}_j)$

$\Leftrightarrow G \prod_{j \in J-S} \bar{F}_j \leq \sum_{S' \subseteq J} \left(\prod_{j \in S'} \bar{B}_j \cdot \prod_{j \in J-S'} \bar{F}_j \right)$

$\Rightarrow G \prod_{j \in J-S} \bar{F}_j \leq \sum_{S \subsetneq S' \subsetneq J} \left(\prod_{j \in S'} \bar{B}_j \prod_{j \in J-S'} \bar{F}_j \right)$

car $G \prod_{j \in J-S} \bar{F}_j \not\leq \sum_{S' \subset S} \left(\prod_{j \in S'} \bar{B}_j \prod_{j \in J-S'} \bar{F}_j \right)$

car pour $S' \subset S$, chaque terme du membre de droite est formé d'un produit contenant plus de variables $\bar{F}_j$ que le membre de gauche.

L'indépendance de G par rapport aux variables $\bar{F}_j$ entraîne :

$$G \leq \sum_{S \subsetneq S' \subsetneq J} \left(\prod_{j \in S'} \bar{B}_j \right) = \prod_{j \in S} \bar{B}_j$$

Remarques :

a) La démonstration ci-dessus fait intervenir des propriétés liées à la notion d'indépendance entre variables que nous renonçons à démontrer dans le cadre de ce travail.

Nous avons en effet admis que :

$YX \leqslant ZX \Longrightarrow Y \leqslant Z$ si Y et Z sont indépendants de X.

b) Si on calcule les impliquants premiers de $[W-]$, de la forme $A \prod_{j \in J-S} \bar{F}_j$, alors tout terme de la forme $A \prod_{j \in J-S'} \bar{F}_j$ avec $S' \supset S$ n'est pas un impliquant de $[W-]$, c'est-à-dire que $\prod_{j \in S' \supset S} \bar{B}_j$ ne couvre pas A.

Tout impliquant premier de $[W-]$, fournit donc les impliquants premiers A du produit des bornes supérieures du plus grand nombre de fonctions.

4.6. Calcul de $[W-]$ par inversion.

Nous avons vu au chap. 2.9. que le calcul des impliquants premiers de la borne supérieure d'une fonction unique se ramène à l'inversion d'une fonction $[Y-]$, comme suit :

- on "masque" les termes de V comprenant les variables F, ceci en effectuant le produit $[V-] = V \cdot \bar{F}$
- on inverse chaque variable de sortie, ce qui donne : $[Y-]$.

Ce procédé peut se généraliser pour les systèmes de plusieurs fonctions :

Nous avons vu aux § 4.2. et 4.3. :

$$V = \sum_{i \in I} E_i S_i \qquad W = \prod_{i \in I} (\bar{E}_i + S_i)$$

on pose : $S_i = \prod_{j \in J} S_{ij}$

où : $S_{ij} = F_j$ si F_j vaut 1 à la ligne i du tableau

$S_{ij} = \bar{F}_j$ si F_j vaut 0 à la ligne i du tableau

Explicitons W :

$$W = \prod_{i \in I} (\bar{E}_i + \prod_{j \in J} S_{ij})$$

$$= \prod_{i \in I} \prod_{j \in J} (\bar{E}_i + S_{ij})$$

$$= \prod_{j \in J} \prod_{i \in I} (\bar{E}_i + S_{ij}) = \prod_{j \in J} W_j$$

avec : $W_j = \prod_{i \in I} (\bar{E}_i + S_{ij})$

Or nous avons défini au § 4.5. :

$$[W-] = \prod_{j \in J} [W_j-]$$

avec : $\left[W_j - \right] = \bar{F}_j + \bar{B}_j = \bar{F}_j + W_j$

car par analogie au chap. 2.9. :

$$W_j = \bar{A}_j \bar{F}_j + \bar{B}_j F_j + \bar{A}_j \bar{B}_j$$

et : $\left[W_j - \right] = \bar{F}_j + \bar{A}_j \bar{F}_j + \bar{B}_j F_j + \bar{A}_j \bar{B}_j = \bar{F}_j + \bar{B}_j$

ainsi $\left[W - \right]$ apparaît sous la forme :

$$\left[W - \right] = \prod_{j \in J} (\bar{F}_j + W_j) = \prod_{j \in J} (\bar{F}_j + \prod_{i \in I} (\bar{E}_i + S_{ij}))$$

$$= \prod_{j \in J} \prod_{i \in I} (\bar{F}_j + \bar{E}_i + S_{ij})$$

$$\left[W - \right] = \prod_{i \in I} (\bar{E}_i + \prod_{j \in J} (\bar{F}_j + S_{ij}))$$

$$\left[W - \right] = \prod_{i \in I} (\bar{E}_i + \left[S_i - \right])$$

En définissant $\left[S_i - \right]$ comme suit :

$$\left[S_i - \right] = \prod_{j \in J} (\bar{F}_j + S_{ij})$$

comme : $\left[W - \right] = \prod_{i \in I} (\bar{E}_i + \left[S_i - \right])$

$$\left[\tilde{W} - \right] = \sum_{i \in I} E_i \left[\bar{S}_i - \right]$$

en posant : $$[V\text{-}] = \sum_{i \in I} E_i [S_i\text{-}]$$

on constate que le procédé est bien la généralisation de celui du § 2.9. :

Le calcul des impliquants premiers de $[W\text{-}]$ peut se faire par inversion d'une fonction

$$[Y\text{-}] = \sum_{i \in I} E_i [\bar{S}_i\text{-}]$$

qui s'obtient comme suit :

- on supprime dans chaque terme de V, les variables F_j non niées. (On remplace S_i par $[S_i\text{-}]$).
- on inverse dans chaque terme de V le produit des variables de sortie. (On remplace $[S_i\text{-}]$ par $[\bar{S}_i\text{-}]$).

4.7. Exemple du § 4.1.

On avait :

$$V = \bar{a}_1\bar{a}_2a_3F_1F_2 + a_1\bar{a}_2\bar{a}_3\bar{F}_1\bar{F}_2 + a_1\bar{a}_2a_3F_1\bar{F}_2 + a_1a_2a_3F_2$$

On en tire :

$$[\bar{W}\text{-}] = \bar{a}_1\bar{a}_2a_3(\bar{1}) + a_1\bar{a}_2\bar{a}_3(\overline{\bar{F}_1\bar{F}_2}) + a_1\bar{a}_2a_3(\bar{F}_2) + a_1a_2a_3(\bar{1})$$

$$= a_1\bar{a}_2\bar{a}_3F_1 + a_1\bar{a}_2\bar{a}_3F_2 + a_1\bar{a}_2a_3F_2$$

En effectuant l'inversion, on trouve :

$$P_1 = \bar{a}_1 + a_2 + a_3 + \bar{F}_1$$

$$P_2 = P_1(\bar{a}_1 + a_2 + a_3 + \bar{F}_2) = \bar{a}_1 + a_2 + a_3 + \bar{F}_1\bar{F}_2$$

$$P_3 = P_2(\bar{a}_1 + a_2 + \bar{a}_3 + \bar{F}_2) = \bar{a}_1 + a_2 + a_3\bar{F}_2 + \bar{F}_1\bar{F}_2$$

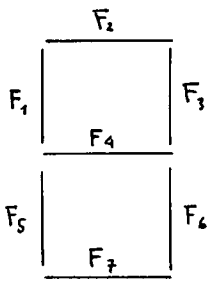
$$[\,W-\,] = \bar{a}_1 + a_2 + a_3\bar{F}_2 + \bar{F}_1\bar{F}_2$$

Commentaire :

$\bar{a}_1$ et a_2 sont impliquants premiers communs de F_1 et F_2.
a_3 est impliquant premier de F_1.
$\bar{F}_1\bar{F}_2$ signifie que 1 n'est impliquant ni de F_1 ni de F_2.

4.8. Exemple du décodeur BCD-7 segments.

La fonction véracité du décodeur BCD-7 segments est la suivante :



$$
\begin{aligned}
V = \quad & \bar{a}\,\bar{b}\,\bar{c}\,\bar{d}\,F_1\,F_2\,F_3\,\bar{F}_4\,F_5\,F_6\,F_7 \\
& + \bar{a}\,\bar{b}\,\bar{c}\,d\,\bar{F}_1\,\bar{F}_2\,F_3\,\bar{F}_4\,\bar{F}_5\,F_6\,\bar{F}_7 \\
& + \bar{a}\,\bar{b}\,c\,\bar{d}\,\bar{F}_1\,F_2\,F_3\,F_4\,F_5\,\bar{F}_6\,F_7 \\
& + \bar{a}\,\bar{b}\,c\,d\,\bar{F}_1\,F_2\,F_3\,F_4\,\bar{F}_5\,F_6\,F_7 \\
& + \bar{a}\,b\,\bar{c}\,\bar{d}\,F_1\,\bar{F}_2\,F_3\,F_4\,\bar{F}_5\,F_6\,\bar{F}_7 \\
& + \bar{a}\,b\,\bar{c}\,d\,F_1\,F_2\,\bar{F}_3\,F_4\,\bar{F}_5\,F_6\,F_7 \\
& + \bar{a}\,b\,c\,\bar{d}\,F_1\,\bar{F}_2\,\bar{F}_3\,F_4\,F_5\,F_6\,F_7 \\
& + \bar{a}\,b\,c\,d\,\bar{F}_1\,F_2\,F_3\,\bar{F}_4\,\bar{F}_5\,F_6\,\bar{F}_7 \\
& + a\,\bar{b}\,\bar{c}\,\bar{d}\,F_1\,F_2\,F_3\,F_4\,F_5\,F_6\,F_7 \\
& + a\,\bar{b}\,\bar{c}\,d\,F_1\,F_2\,F_3\,F_4\,\bar{F}_5\,F_6\,\bar{F}_7
\end{aligned}
$$

Nous allons calculer les impliquants premiers de ce système par le processus d'inversion de la fonction unique $[\,\bar{W}\text{-}\,] = \sum_{i \in I} E_i\,[\,\bar{S}_i\text{-}\,]$.

La fonction véracité $[\,\bar{W}\text{-}\,]$, après développement en somme de produits, apparaît comme suit :

$$
\begin{aligned}
[\,\bar{W}\text{-}\,] = \quad & \bar{a}\,\bar{b}\,\bar{c}\,\bar{d}\,F_4 + \bar{a}\,\bar{b}\,\bar{c}\,d\,F_1 + \bar{a}\,\bar{b}\,\bar{c}\,d\,F_2 \\
& + \bar{a}\,\bar{b}\,\bar{c}\,d\,F_4 + \bar{a}\,\bar{b}\,\bar{c}\,d\,F_5 + \bar{a}\,\bar{b}\,\bar{c}\,d\,F_7 \\
& + \bar{a}\,\bar{b}\,c\,\bar{d}\,F_1 + \bar{a}\,\bar{b}\,c\,\bar{d}\,F_6 + \bar{a}\,\bar{b}\,c\,d\,F_1 \\
& + \bar{a}\,\bar{b}\,c\,d\,F_5 + \bar{a}\,b\,\bar{c}\,\bar{d}\,F_2 + \bar{a}\,b\,\bar{c}\,\bar{d}\,F_5 \\
& + \bar{a}\,b\,\bar{c}\,\bar{d}\,F_7 + \bar{a}\,b\,\bar{c}\,d\,F_3 + \bar{a}\,b\,\bar{c}\,d\,F_5 \\
& + \bar{a}\,b\,c\,\bar{d}\,F_2 + \bar{a}\,b\,c\,\bar{d}\,F_3 + \bar{a}\,b\,c\,d\,F_1 \\
& + \bar{a}\,b\,c\,d\,F_4 + \bar{a}\,b\,c\,d\,F_5 + \bar{a}\,b\,c\,d\,F_7 \\
& + a\,\bar{b}\,\bar{c}\,d\,F_5 + a\,\bar{b}\,\bar{c}\,d\,F_7
\end{aligned}
$$

Et le processus d'inversion $[\bar{W}\text{-}] \rightarrow [W\text{-}]$ donne finalement :

$$
\begin{aligned}
[W\text{-}] = \quad & \bar{F}_1\ \bar{F}_2\ \bar{F}_3\ \bar{F}_4\ \bar{F}_5\ \bar{F}_6\ \bar{F}_7 \\
& + \bar{b}\ \bar{c}\ \bar{F}_1\ \bar{F}_2\ \bar{F}_4\ \bar{F}_5\ \bar{F}_7 \\
& + \bar{c}\ \bar{F}_1\ \bar{F}_2\ \bar{F}_3\ \bar{F}_4\ \bar{F}_5\ \bar{F}_7 \\
& + \bar{b}\ d\ \bar{F}_1\ \bar{F}_2\ \bar{F}_4\ \bar{F}_5\ \bar{F}_7 \\
& + d\ \bar{F}_1\ \bar{F}_2\ \bar{F}_3\ \bar{F}_4\ \bar{F}_5\ \bar{F}_7 \\
& + \bar{b}\ \bar{F}_1\ \bar{F}_2\ \bar{F}_4\ \bar{F}_5\ \bar{F}_6\ \bar{F}_7 \\
& + \bar{b}\ \bar{c}\ \bar{d}\ \bar{F}_4 \\
& + \bar{b}\ \bar{d}\ \bar{F}_1\ \bar{F}_4\ \bar{F}_6 \\
& + \bar{c}\ \bar{d}\ \bar{F}_2\ \bar{F}_4\ \bar{F}_5\ \bar{F}_7 \\
& + \bar{b}\ c\ \bar{d}\ \bar{F}_1\ \bar{F}_6 \\
& + \bar{b}\ c\ \bar{F}_1\ \bar{F}_5\ \bar{F}_6 \\
& + b\ \bar{c}\ \bar{d}\ \bar{F}_2\ \bar{F}_5\ \bar{F}_7 \\
& + c\ \bar{d}\ \bar{F}_1\ \bar{F}_2\ \bar{F}_3\ \bar{F}_6 \\
& + b\ \bar{c}\ d\ \bar{F}_3\ \bar{F}_5 \\
& + b\ \bar{c}\ \bar{F}_2\ \bar{F}_3\ \bar{F}_5\ \bar{F}_7 \\
& + b\ \bar{d}\ \bar{F}_2\ \bar{F}_3\ \bar{F}_5\ \bar{F}_7 \\
& + b\ c\ \bar{d}\ \bar{F}_2\ \bar{F}_3 \\
& + \bar{b}\ c\ d\ \bar{F}_1\ \bar{F}_5 \\
& + b\ \bar{F}_1\ \bar{F}_2\ \bar{F}_3\ \bar{F}_4\ \bar{F}_5\ \bar{F}_7
\end{aligned}
$$

$$
\begin{aligned}
& + c\ d\ \bar{F}_1\ \bar{F}_4\ \bar{F}_5\ \bar{F}_7 \\
& + b\ d\ \bar{F}_1\ \bar{F}_3\ \bar{F}_4\ \bar{F}_5\ \bar{F}_7 \\
& + a\ b \\
& + a\ c \\
& + a\ \bar{d} \\
& + a\ \bar{F}_5\ \bar{F}_7
\end{aligned}
$$

Cet exemple bien connu et qui est déjà trop complexe pour être calculé à la main, se laisse résoudre en un temps minime sur les plus petits systèmes programmables.

CHAPITRE 5. CALCUL DES IMPLIQUANTS PREMIERS D'UN SYSTEME DE FONCTIONS PAR UN ALGORITHME D'INVERSION SIMULTANEE.

5.1. Introduction.

Nous avons vu au chapitre 4, que les impliquants premiers d'un système de fonctions s'obtiennent par le processus d'inversion de la fonction unique $[\bar{W}-] = \sum_{i \in I} E_i [\bar{S}_i-]$. La nécessité de devoir développer chaque terme $E_i [\bar{S}_i-]$ en une somme de termes:

$$E_i [\bar{S}_i-] = E_i [\bar{S}_{i1}-] + E_i [\bar{S}_{i2}-] + \ldots$$

comporte deux inconvénients :

- elle augmente le nombre de termes de $[\bar{W}-]$
- elle ralentit le processus d'inversion, puisqu'il faut effectuer une itération par terme.

Nous allons voir comment il est possible d'accélérer le processus d'inversion, tout en diminuant le nombre de termes à mémoriser.

5.2. Principe de l'algorithme d'inversion simultanée.

$[W-]$ s'obtient par inversion de :

$$[\bar{W}-] = \sum_{i \in I} E_i [\bar{S}_i-]$$

Au lieu de développer les $E_i\,[\bar{S}_i-]$ en sommes de produits, introduisons-les tels quels dans le processus itératif d'inversion :

$$P_0 = 1$$

$$P_1 = P_0(\bar{E}_1 + [S_1-]) = \bar{E}_1 + [S_1-]$$

$$P_2 = P_1(\bar{E}_2 + [S_2-]) = \ldots$$

$$\vdots$$

$$P_I = P_{I-1}(\bar{E}_I + [S_I-])$$

A chaque itération, on effectue :

$$P_i = P_{i-1}(\bar{E}_i + [S_i-])$$

Or P_{i-1} est déjà sous la forme d'une somme de produits de la forme :

$$P_{i-1} = \sum_k B_k\, S_k$$

avec B_k ne dépendant que des variables d'entrée et S_k ne dépendant que des variables de sortie.

En développant P_i, on obtient :

$$P_i = (\sum_k B_k\, S_k)(\bar{E}_i + [S_i-])$$

$$P_i = \sum_k (B_k\, \bar{E}_i)\, S_k + \sum_k B_k (S_k\, [S_i-])$$

On constate que le calcul de :

$$\sum_k (B_k \bar{E}_i) S_k$$

correspond au calcul effectué dans l'algorithme d'inversion d'une fonction simple.

D'autre part, le calcul de :

$$\sum_k B_k (S_k [S_i-])$$

ne nécessite que la manipulation des variables $\bar{F}_j$ dont dépendent S_k et $[S_i-]$.

L'algorithme d'inversion simultané peut donc s'énoncer :

A partir du "tableau source" P_{i-1}, former le tableau P_i comme suit :

1) Développer chaque terme $B_k S_k$ du tableau source selon l'algorithme d'inversion en ne considérant que les variables d'entrée.

   Affecter à chaque ligne obtenue le produit des variables de sortie de la ligne source.

2) Ajouter le "tableau source" dans le tableau produit, après avoir multiplié le produit S_k des variables de sortie de chaque ligne source par le produit des variables de sortie $[S_i-]$ de la ligne $E_i S_i$.

5.3. Efficacité de l'algorithme d'inversion simultanée.

Estimons le nombre de produits logiques qu'il faut effectuer lors de l'inversion $[\bar{W}] \rightarrow [W]$.

Soit un tableau de I lignes comportant chacune N variables d'entrées et J variables de sortie.

Dans le cas où on développe $[\bar{W}]$ en somme de produits (chapitre 4.5.), chaque ligne $E_i [\bar{S}_i]$ donne un nombre de termes de N + 1 variables, qui croît comme J.

L'inversion du tableau entier nécessite un nombre de produits qui croît comme :

$$I \cdot J \cdot (N + 1) \text{ produits.}$$

Pour l'algorithme simultané, dans lequel on traîte séparément entrées et sorties, l'inversion nécessite le produit de I lignes contenant J + N variables, ce qui donne :

$$I \cdot (J + N) \text{ produits.}$$

Pour $J \gg 1$, la seconde solution est visiblement avantageuse !

Pour J = 1, les 2 solutions sont équivalentes.

En conclusion, il faut remarquer que le nombre $I \cdot (J + N)$ représente le nombre de bits définis dans le tableau décrivant le système.

5.4. Exemple du § 4.1.

On avait :

$$[V-] = a_1\bar{a}_2\bar{a}_3\bar{F}_1\bar{F}_2 + a_1\bar{a}_2a_3\bar{F}_2$$

$$P_0 = 1$$

$$P_1 = P_0(\bar{a}_1 + a_2 + a_3) + P_0\bar{F}_1\bar{F}_2 = \bar{a}_1 + a_2 + a_3 + \bar{F}_1\bar{F}_2$$

$$P_2 = P_1(\bar{a}_1 + a_2 + \bar{a}_3) + P_1\bar{F}_2 = \bar{a}_1 + a_2 + \bar{a}_1F_2 + a_2\bar{F}_2 + a_3\bar{F}_2 + \bar{F}_1\bar{F}_2$$

$$[W-] = \bar{a}_1 + a_2 + a_3\bar{F}_2 + \bar{F}_1\bar{F}_2$$

Ce qui est bien identique au résultat obtenu sous 4.7.

CHAPITRE 6. APPLICATION DE L'ALGORITHME D'INVERSION SIMULTANEE AU CALCUL DES COUVERTURES PREMIERES DES SYSTEMES DE FONCTIONS.

6.1. Définition du problème.

Nous avons vu aux chapitres 4 et 5 comment obtenir les impliquants premiers d'un système de fonctions par l'inversion d'une fonction booléenne $[\bar{W}-]$ obtenue à partir du tableau décrivant le système. Les impliquants de $[W-]$ ont la forme :

$$H_k = G_k \prod_{j \in J-S_k} \bar{F}_j \leq [W-]$$

où : $k \in K$ nombre d'impliquants premiers de $[W-]$

$S_k \subseteq J$ sous-ensemble des indices des fonctions F_j, pour lesquelles G_k n'est pas impliquant.

G_k ne dépend que des variables d'entrée.

Le problème consiste à trouver pour chaque F_j une somme partielle de G_k, $k \in K_j \subseteq K$ couvrant F_j, de manière à ce que la réunion des K_j comporte un minimum d'éléments.

C'est-à-dire :

$\forall$ j on cherche K_j tel que : $A_j \leq \sum_{k \in K_j} G_k$

A_j défini par $V_j = A_j F_j + B_j \bar{F}_j$

et : $\bigcup_{j \in J} K_j$ comportant le minimum de termes.

Les deux conditions ci-dessus font apparaître le caractère simultané de la recherche des ensembles K_j.

6.2. Description de la méthode.

La méthode de calcul des couvertures premières d'un système de fonctions est dans son principe la généralisation de la méthode utilisée pour un système à sortie unique démontrée au chap. 3.

Donnons-en une description suivie d'un exemple, sans toutefois la démontrer.

Nous disposons d'un ensemble de K impliquants de $[W-]$ de la forme :

$$H_k = G_k \prod_{j \in J-S_k} \bar{F}_j$$

dont on peut dire que G_k n'est pas impliquant des fonctions F_j, j appartenant à $J-S_k$.

On forme l'expression correspondante :

$$H_k^* = G_k \sum_{j \in S_k} F_j$$

qui signifie que G_k peut être utilisé pour couvrir l'une ou l'autre des F_j, $j \in S_k$.

D'autre part, à partir de :

$$V = \sum_{i \in I} E_i S_i = \sum_{i \in I} \left(E_i \prod_{j \in T_i} F_j \prod_{j \in U_i} \bar{F}_j \right)$$

avec : $T_i \cup U_i \subseteq J$

et : $T_i \cap U_i = \{0\}$

on forme la fonction :

$$[V+]^* = \sum_{i \in I} E_i \prod_{j \in J - U_i} \bar{F}_j$$

qui est "une transformée" de la fonction véracité de la borne inférieure du système.

La recherche d'une couverture irrédondante de la borne inférieure du système se ramène à la recherche d'une somme partielle H_S^*.

$$H_S^* = \sum_{k \in S} H_k^* \quad \text{avec } S \subseteq K.$$

vérifiant la relation :

$$[V+]^* \leq H_S^*$$

Pour ce faire, on forme la fonction :

$$H_S^*(X,F) = \sum_{k \in K} H_k^* \, X_k$$

et on calcule les impliquants premiers de la fonction véracité :

$$\bar{R}(X,F) = \left[V+ \right]^* \cdot \overline{H_S^*(X,F)}$$

On obtient $\bar{R}(X,F)$ sous la forme :

$$\bar{R}(X,F) = \sum_{l \in L} B_l \prod_{k \in K_l} \bar{X}_k \prod_{j \in J_l} \bar{F}_j \qquad \text{où : } K_l \leq K \quad J_l \leq J$$

Il est alors nécessaire de transformer $\bar{R}(X,F)$ en $\bar{R}^*(X,F)$ défini comme suit :

$$\bar{R}^*(X,F) = \sum_{l \in L} B_l \prod_{k \in K_l} \bar{X}_k \sum_{j \in J-J_l} F_j$$

Il reste à effectuer le calcul de $R^*(X,F)$ en inversant l'expression ci-dessus :

$$R^*(X,F) = \prod_{l \in L} \left(\bar{A}_l + \sum_{k \in K_l} X_k + \prod_{j \in J-J_l} \bar{F}_j \right)$$

Les solutions de cette expression ne dépendant que de X et de F, s'obtiennent par le calcul de l'expression simplifiée:

$$R^*(X,F) = \prod_{l \in L} \left(\sum_{k \in K_l} X_k + \prod_{j \in J-J_l} \bar{F}_j + \ldots \right)$$

6.3. Interprétation de R*(X,F).

L'interprétation de R*(X,F) simplifié est liée au théorème suivant :

$$A_j \leq \sum_{k \in S} G_k \quad \forall\, j \in T \quad \Longleftrightarrow \quad \prod_{k \in S} X_k \prod_{j \in J-T} \bar{F}_j \leq R^*(X,F)$$

Ce qui s'énonce :

A un impliquant premier de R*(X,F) de la forme $\prod_{k \in S} X_k \prod_{j \in J-T} \bar{F}_j$, correspond la somme des impliquants $\sum_{k \in S} G_k$ qui couvrent les bornes inférieures A_j des fonctions F_j, $j \in T$, de manière irrédondante.

Nous admettons sans démonstration ce théorème, qui est une généralisation des théorèmes 5 (chap. 3.2.) et 7 (chap. 4.5.).

Cas particulier :

A tout impliquant de R*(X,F) de la forme :

$$X_S = \prod_{k \in S} X_k$$

correspond un ensemble de G_k, $k \in S$, couvrant le système intégralement.

A tout impliquant de R*(X,F) de la forme :

$$\prod_{k \in S'} X_k \prod_{j \in J-\{j_0\}} \bar{F}_j \qquad \text{avec } S' \subseteq S$$

correspond le sous-ensemble des G_k, $k \in S'$, couvrant la fonction F_{j_0} uniquement.

Ce qui signifie que R*(X,F) fournit non seulement les solutions irrédondantes du système complet, mais encore les solutions irrédondantes de tous les sous-systèmes, notamment les solutions irrédondantes de chaque fonction individuelle.

6.4. Interprétation de $\bar{R}(X,F)$.

Nous avons vu que les impliquants de $\bar{R}(X,F)$:

$$B_1 \prod_{k \in K_1} \bar{X}_k \prod_{j \in J_1} \bar{F}_j$$

peuvent s'interpréter en considérant le théorème :

$$\prod_{k \in K_1} G_k \cdot B_1 \cdot \prod_{j \in J-J_1} A_j \neq 0 \Longrightarrow B_1 \prod_{k \in K_1} \bar{X}_k \prod_{j \in J_1} \bar{F}_j < \bar{R}(X,F)$$

C'est-à-dire qu'un impliquant de $\bar{R}(X,F)$ de la forme ci-dessus signifie que B_1 est couvert, au moins partiellement, par chaque G_k, $k \in K_1$ et ceci pour chaque fonction F_j, $j \in J\text{-}J_1$.

Ce théorème que nous admettons également sans démonstration (et sous toute réserve) représente la généralisation des théorèmes 6 (chap. 3.4.1.) et 7 (chap. 4.5.).

6.5. Exemple du chapitre 4.1.

La fonction véracité du système est :

$$V = \bar{a}_1\bar{a}_2a_3F_1F_2 + a_1\bar{a}_2\bar{a}_3\bar{F}_1\bar{F}_2 + a_1\bar{a}_2a_3F_1\bar{F}_2 + a_1a_2a_3F_2$$

dont nous avons trouvé les impliquants premiers. (chap. 5.5.).

$$[W\text{-}] = \bar{a}_1 + a_2 + a_3\bar{F}_2 + \bar{F}_1\bar{F}_2$$

On forme la fonction :

$$H(X,F)^* = \bar{a}_1(F_1 + F_2)\,X_1 + a_2(F_1 + F_2)\,X_2 + a_3F_1X_3$$

et la fonction :

$$[V+]^* = \bar{a}_1\bar{a}_2a_3 + a_1\bar{a}_2\bar{a}_3\bar{F}_1\bar{F}_2 + a_1\bar{a}_2a_3\bar{F}_2 + a_1a_2a_3\bar{F}_1$$

et on calcule :

$$\bar{R}\ (X,F) = \left[V+\right]^* \cdot \overline{H^*(X,F)}$$

d'où :

$$p_0 = \bar{a}_1\bar{a}_2a_3 + a_1\bar{a}_2\bar{a}_3\bar{F}_1\bar{F}_2 + a_1\bar{a}_2a_3\bar{F}_2 + a_1a_2a_3\bar{F}_1$$

$$p_1 = P_0(a_1+ \bar{X}_1+ \bar{F}_1\bar{F}_2) = \bar{a}_1\bar{a}_2a_3\bar{X}_1 + \bar{a}_1\bar{a}_2a_3\bar{F}_1\bar{F}_2 + a_1\bar{a}_2\bar{a}_3\bar{F}_1\bar{F}_2 + a_1\bar{a}_2a_3\bar{F}_2 + a_1a_2a_3\bar{F}_1$$

$$p_2 = P_1(\bar{a}_2+ \bar{x}_2+ \bar{F}_1\bar{F}_2) = \bar{a}_1\bar{a}_2a_3\bar{X}_1 + \bar{a}_1\bar{a}_2a_3\bar{F}_1\bar{F}_2 + a_1\bar{a}_2\bar{a}_3\bar{F}_1\bar{F}_2 + a_1\bar{a}_2a_3\bar{F}_2 + a_1a_2a_3\bar{X}_2\bar{F}_1 + a_1a_2a_3\bar{F}_1\bar{F}_2$$

$$p_3 = P_2(\bar{a}_3+ \bar{X}_3+ \bar{F}_1) = \bar{a}_1\bar{a}_2a_3\bar{X}_1\bar{X}_3 + \bar{a}_1\bar{a}_2a_3\bar{X}_1\bar{F}_1 + \bar{a}_1\bar{a}_2a_3\bar{F}_1\bar{F}_2 + a_1\bar{a}_2\bar{a}_3\bar{F}_1\bar{F}_2 + a_1\bar{a}_2a_3\bar{X}_3\bar{F}_2 + a_1\bar{a}_2a_3\bar{F}_1\bar{F}_2 + a_1a_2a_3\bar{X}_2\bar{F}_1 + a_1a_2a_3\bar{F}_1\bar{F}_2$$

d'où :

$$\bar{R}(X,F) = \bar{a}_1\bar{a}_2a_3\bar{X}_1\bar{X}_3 + \bar{a}_1\bar{a}_2a_3\bar{X}_1\bar{F}_1 + a_1\bar{a}_2a_3\bar{X}_3\bar{F}_2 + a_1a_2a_3\bar{X}_2\bar{F}_1 + \ldots$$

Interprétation :

- Le terme $\bar{a}_1\bar{a}_2a_3\bar{X}_1\bar{X}_3$ signifie que $\bar{a}_1\bar{a}_2a_3$ a une intersection non nulle (couvert dans ce cas, car c'est un minterme) avec les impliquants premiers $\bar{a}_1$ indicé par X_1 et a_3 indicé par X_3, et que ces deux derniers ne couvrent aucun zéro des fonctions F_1 et F_2 de la ligne $\bar{a}_1\bar{a}_2a_3F_1F_2$. Plus simplement, si on suppose $\bar{a}_1\bar{a}_2a_3 = 1$, alors il suffit que $\bar{X}_1\bar{X}_3 = 1$ pour que $\bar{R}(X,F) = 1$, c'est-à-dire pour que la condition de couverture ne soit pas remplie à cette ligne.

  Réciproquement, il suffit que $\bar{X}_1\bar{X}_3 = 0$, c'est-à-dire que $X_1 = 1$ ou $X_3 = 1$ pour que F_1 et F_2 soient couverts à cette ligne.

- Le terme $\bar{a}_1\bar{a}_2a_3\bar{X}_1\bar{F}_1$ signifie que X_1 doit être égal à 1 pour que F_2 soit couvert à cette ligne. En effet, $a_3\bar{F}_2 \leq [W-]$ signifie que a_3 n'est pas impliquant de F_2.

- Les termes tels que $\bar{a}_1\bar{a}_2a_3\bar{F}_1\bar{F}_2$, desquels les $\bar{X}_k$ sont absents et qui contiennent (par conséquent) le produit de tous les $\bar{F}_j$, $j \notin J$, ne contiennent aucune information utile. Il sont d'ailleurs éliminés par l'opération $\bar{R} \rightarrow \bar{R}^*$ effectuée ci-dessous.

A partir de $\bar{R}(X,F)$, formons la fonction :

$$\bar{R}^*(X,F) = \bar{a}_1\bar{a}_2a_3\bar{X}_1\bar{X}_3(F_1+F_2) + \bar{a}_1\bar{a}_2a_3\bar{X}_1F_2$$
$$+ a_1\bar{a}_2a_3\bar{X}_3F_1 + a_1a_2a_3\bar{X}_2F_2 + \bar{a}_1\bar{a}_2a_3(0) + \ldots$$

En inversant cette dernière expression et en ne considérant que les termes ne dépendant que de X et de F, nous obtenons :

$$R^* = (X_1+X_3+\bar{F}_1\bar{F}_2)(X_1+\bar{F}_2)(X_3+\bar{F}_1)(X_2+\bar{F}_2)$$

$$R^* = X_1X_2X_3 + X_1X_2\bar{F}_1 + X_3\bar{F}_2$$

Interprétation :

- $X_1X_2X_3$ signifie que les impliquants $\bar{a}_1$, a_2, a_3 indicés respectivement par X_1, X_2 et X_3, sont un couverture irrédondante (la seule dans ce cas) du système.

- $X_1X_2\bar{F}_1$ signifie que F_2 est couvert de manière irrédondante par $\bar{a}_1$ et a_2.

- d'où la solution :

$$F_1 = a_3$$

$$F_2 = \bar{a}_1 + a_2$$

6.6. Conclusion.

Les transformations du type $H_k \longrightarrow H_k^*$ apparaissant dans la description de ce procédé paraissent "artificielles". La nécessité de leur introduction s'explique par le fait que l'algorithme d'inversion simultanée exposé au chapitre 5., a été conçu de manière à ne manipuler que des expressions contenant des produits de variables de sorties niées, les autres n'étant pas prises en considération dans le processus. Dans l'exemple ci-dessus, on constate effectivement que le calcul de $\bar{R}(X,F)$ et de $R^*(X,F)$ ne s'effectue que sur des expressions de ce type.

Du point de vue de la réalisation programmée de la méthode, l'introduction de ce type de transformation est simple et naturelle dans le cas de cette conception de l'algorithme d'inversion simultanée.

CONCLUSION.

L'introduction de variables caractéristiques liées aux entrées et aux sorties d'un système combinatoire, ainsi que des fonctions véracité, caractérisant les relations booléennes entre celles-ci, permet d'effectuer toutes les opérations habituelles de la synthèse en sommes de produits à l'aide d'un seul et même algorithme. De plus, on a montré que, grâce à cette méthode, la manipulation explicite des variables et des états non définis du système disparaît, ce qui n'est pas le cas pour les méthodes utilisant la notation "cubique". En effet, dans cette dernière notation, on écrirait l'impliquant $A\bar{D}G$ d'une fonction F(A,B,C,D,E,F,G,H,I) sous la forme 1--0--1--. Dans ce cas, c'est la position dans un "mot machine", de longueur fixe et dépendant du nombre de variables de la fonction, qui est caractéristique de chaque variable du système, ce qui explique que les programmes réalisés dans cette technique ne peuvent traiter généralement des fonctions dépendant d'un nombre de variables proportionnel au nombre de bits du mot de la machine utilisée.

Dans notre méthode, on ne code qu'une seule variable caractéristique dans un mot machine, et les variables définies d'un produit sont stockées de manière contiguë entre deux séparateurs en mémoire.

On peut ainsi traiter des problèmes dépendant d'un nombre de variables qui croît exponentiellement avec le nombre de bits du mot de la machine utilisée. On ne peut alors plus effectuer le produit logique de deux impliquants en parallèle pour toutes les variables, mais il faut effectuer ce produit séquentiellement. Cet inconvénient n'est cependant pas gênant, car le classement par ordre alphabétique des variables en mémoire ainsi que le fait que les états indéfinis de celles-ci ne sont pas codées, a permis de développer une routine de produit de deux impliquants dont le nombre d'opérations dépend linéairement du nombre de variables définies (cf. annexe 1.).

C'est ce qui a notamment permis d'introduire les variables auxiliaires d'optimisation, dont le nombre est égal à celui des impliquants premiers d'un système, et ainsi de supprimer la manipulation des tables de couvertures.

La méthode a été programmée sur une calculatrice de table et permet de traiter des systèmes jusqu'à vingt à trente variables dans des temps raisonnables. Un programme plus élaboré (cf. annexe 1.), a été réalisé sur un système à microprocesseur 8-bits.

Celui-ci s'avère actuellement très utile pour la synthèse des PLAs utilisées dans les circuits à haute densité d'intégration. On atteint une réduction de la surface d'environ 20 %, alors qu'on peut s'attendre à une réduction de la consommation encore supérieure !

Enfin, la démonstration de l'algorithme de calcul simultané des impliquants premiers d'un système, ainsi que celle du calcul des couvertures irrédondantes par l'intermédiaire de la recherche des solutions d'une équation booléenne du type $R^*(X,F) = 1$ sont originales et devraient apporter une contribution à l'unification des formalismes dans la synthèse des systèmes combinatoires.

REMERCIEMENTS.

Mes remerciements chaleureux vont à M. F. Pellandini pour la compréhension et le soutien qu'il m'a apporté par son analyse critique et les encouragements qu'il m'a prodigués tout au long de la rédaction de ce travail. A M. J.C. Zahnd, dont la contribution a été essentielle à la clarification des théorèmes de couverture et de l'ensemble du formalisme développé. A MM. A. Robert et A. Shah, dont les suggestions constructives ont été très utiles.

Enfin, je ne saurais oublier l'aide fraternelle et dynamique que M. N. Péguiron m'a apportée pendant la mise au point du manuscrit, ainsi que la collaboration efficace de M. J.Cl. Blisse dans le domaine de la programmation de la méthode sur microprocesseur.

La recherche faisant l'objet de cette thèse a été financée par le Fonds National pour la Recherche Scientifique Suisse, dans le cadre des projets No 2.558-0.76 et No 2.092-0.78 .

BIBLIOGRAPHIE.

(1) W.V. QUINE : "THE PROBLEM OF SIMPLIFYING TRUTH FUNCTIONS" AM MATH. MONTHLY VOL. 59 OCT. 52 P. 521-531

(2) W.V. QUINE : "A WAY TO SIMPLIFY TRUTH FUNCTIONS" AM MATH. MONTHLY VOL. 62 NOV. 1955 P. 627-631

(3) M.A. GAVRILOV : "MINIMIZATION TO THE BOOLEEAN FUNCTIONS CHARACTERISING SWITCHING CIRCUITS" IRE TRANSACTIONS ON ELECTRONIC COMPUTERS EC 9 DEC. 1960 P. 517

(4) E.J. MC CLUSKEY : "MINIMIZATION OF BOOLEEAN FUNCTIONS" BELL SYSTEM TECHNICAL JOURNAL NO. 6 NOV. 1956 P. 1417-1444

(5) KAZAKOF : "THE MINIMIZATION OF A LARGE NUMBER OF VARIABLES" AUTOMATION AND REMOTE CONTROL MARCH 1967

(6) T.C. BARTEE : "COMPUTER DESIGN OF MULTIPLE OUTPUT DIGITAL NETWORKS" IRE TRANS. ON ELECTRONIC COMPUTERS VOL. EC-10 MARCH 1961 P. 21-30

(7) M. KARNAUGH : "THE MAP METHOD FOR SYNTHESIS OF CONBINATIONAL FOGIC CIRCUITS" TRANS. AIEEE 72 PT.I P. 593-598 1953

(8) J. FLORINE : "LA SYNTHESE DES MACHINES LOGIQUES" DUNOD PARIS 1964

(9) D. MANGE : "CALCULATRICE SPECIALISEE PIM 4 POUR LA SIMPLIFICATION AUTOMATIQUE DES FONCTIONS LOGIQUES" COMMUNICATION DU GROUPEMENT POUR L'ETUDE DES TELECOMMUNICATIONS DE LA FONDATION HASLER BERNE (AGEN) NO.7 1967

(10) P.P. PARKHOMENKO : "MACHINE ANALYSIS OF SWITCHING CIRCUITS" AUTOMATION AND REMOTE CONTROL VOL. 20 1960 JANUARY

(11) F.J. HILL : G.R. PETERSON "INTRODUCTION TO SWITCHING THEORY AND LOGICAL CESIGN" WILEY NEW YORK 1968

(12) D.L. DIETMEYER : "LOGIC DESIGN OF DIGITAL SYSTEMS" ALLYN AND BACON BOSTON 1971

(13) D. LEWIN : "COMPUTER AIDED DESIGN OF DIGITAL SYSTEMS" CRANE RUSSAK NEW YORK 1977

(14) K.S. MENGER : " A TRANSFORM FOR LOGIC NETWORKS" IEEE TRANSACTIONS ON COMPUTERS VOL C-18 MARCH 1969

(15) A. THAYSE : M. DAVIO "BOOLEEAN DIFFERENTIAL CALCULUS AND ITS APPLICATION TO SWITCHING THEORY" IEEE TRANSACTIONS ON COMPUTERS VOL. C-22 NO. 4 APRIL 1973

(16) S.J. HONG : R.G. CAIN D.L. OSTAPKO "MINI : A HEURISTIC APPROACH FOR LOGIC MINIMIZATION" IBM JOURNAL ON RESEARCH AND DEVELOPMENT SEPT. 1974

(17) Z. KOHAVI : "SWITCHING AND FINITE AUTOMATA THEORY" MAC GRAW HILL NEW YORK 1970

(18) P. TISON : "THEORIE DES CONSENSUS" THESE UNIV. DE GRENOBLE 1965

(19) J. KUNTZMANN : "ALGEBRE DE BOOLE" DUNOD PARIS 1968

(20) J.J. MONBARON : "OPTIMISATION DU SYSTEME LOGIQUE DE LA COMMANDE DE L'AFFICHAGE D'UNE MONTRE" TRAV. DE DIPLOME UNIV. DE NEUCHATEL 1973

(21) T.L. DOLLHOFF : B.L. WEINBERG "A RESULT ON SET EXTRACTION AND APPLICATION TO COVERING CLOSURE TABLE" IEEE TRANS. ON COMPUTERS VOL. C-21 P. 603-606 1972

(22) A. DUNWORTH : A. VAN DES KNAPP "AN EFFICIENT IMPLEMENTATION OF THE SHARP PRODUCT OPERATION FOR MULTIPLE OUTPUT SWITCHING CIRCUITS" DIGITAL PROCESSES VOL. 3 P. 161-176 1977

(23) J. FLORINE : "AUTOMATISMES A SEQUENCES ET COMMANDES NUMERIQUES" DUNOD PARIS 1969

(24) T.L. DOLLHOFF : "PROGRAM CUTS LOGIC DESIGN COST" ELECTRONIC DESIGN VOL 22 No. 9 April 74 P. 186-189

(25) A. THAYSE : "LA DETECTION DES ALEAS DANS LES CIRCUITS LOGIQUES AU MOYEN DU CALCUL DIFFERENTIEL BOOLEEN" DIGITAL PROCESSES VOL 1 No. 2 1975 P. 141-169

(26) L. HELLERMAN : "A CATALOG OF THREE-VARIABLE OR-INVERT AND AND-INVERT LOGICAL CIRCUITS" IEEEE TRANSACTIONS ON ELECTRONIC COMPUTERS EC 12 1963 P. 198-223

(27) I. LAVALLEE : "UN ALGORITHME DE DETERMINATION DES COUVERTURES DE CARDINAL MINIMAL" REVUE FRANCAISE D'AUTOMATIQUE, INFORMATIQUE ET RECHERCHE OPERATIONNELLE AOÛT 73 R-2

(28) B. BENJAUTHRIT : I.S. REED "GALOIS SWITCHING FUNCTIONS AND THEIR APPLICATIONS" IEEE TRANSACTIONS ON COMPUTERS VOL. C-25 NO. 1 JANUARY 1976

(29) S. RUDEANU : "SQUARE ROOTS AND FUNCTIONAL DECOMPOSITION" IEEE TRANSACTIONS ON COMPUTERS VOL. C-25 NO. 5 MAY 1976

(30) R.A. SMITH : "MINIMAL THREE VARIABLE NOR AND NAND LOGIC CIRCUITS" IEEE TRANSACTIONS ON ELECTRONIC COMPUTERS VOL EC-14 FEBRUARY 69 P. 79-81

(31) C.R. BAUGH : C.S. CHANDERSEKARAN R.S. SWEE SABURO MUROGA "OPTIMAL NETWORKS OF NOR-OR GATES FOR FUNCTIONS OF THREE VARIABLES" IEEE TRANSACTIONS ON COMPUTERS VOL. C-21 NO. 2 FEBRUARY 1972

(32) E. CERNY : M.A. MARIN "AN APPROACH TO UNIFIED METHODOLOGY OF COMBINATIONAL SWITCHING CIRCUITS" IEEE TRANSACTIONS ON COMPUTERS VOL. C-26 AUGUST 1977 P. 745-756

(33) J. KOZLOWSKI :
E. CERNY

"SUBROUTINES FOR BOOLEEAN ARRAY OPERATIONS" LOCAL REPORT DEPARTMENT E.E. CONCORDIA UNIVERSITY MONTREAL FALL 1976 1977

ANNEXE 1.

La réalisation des programmes sur microprocesseur a été effectuée en collaboration avec Monsieur J. Cl. Blisse. Cette partie est un extrait du travail de certificat intitulé "PROGRAMMATION DES ALGORITHMES DE LA SYNTHESE DES SYSTEMES LOGIQUES SUR MICROPROCESSEUR".

Description des données.

Les données consistent en tableaux de variables logiques. Seuls les états définis du système sont représentés.

Une variable logique est une lettre de a à z, suivie d'un indice de 0 à 9 (par défaut 0), dans le cas du premier codage (Type 1). Il y a donc à disposition 26 * 10 = 260 variables distinctes.

Dans le cas du second codage (Type 2), qui permet un plus grand nombre de variables, mais une moins bonne lisibilité, une variable logique consiste en une lettre quelconque suivie d'un nombre de 0 à 32511 (par défaut 0). La lettre dans ce cas ne sert que de séparateur.

Une variable de type 1 ou 2 peut être suivie d'un apostrophe, pour spécifier qu'elle est niée : a'= non a.

Un tableau est donc une somme de lignes, qui sont elles-mêmes un produit de variables logiques.

Chaque ligne est décomposée en variables d'entrée et variables de sortie, qui sont séparées les unes des autres par un space.

Un carriage-return-line-feed (retour de chariot) termine chaque ligne.

Les données sont donc crées comme n'importe quel fichier source, grâce à l'éditeur du système.

Pour la mise en page, on peut intercaler un ou plusieurs controle-I (tabulateur) avant les variables d'entrée et entre celles-ci et les variables de sortie. Le space séparant variables d'entrée et de sortie peut être entouré ou non de controle-I.

Enfin, les noms de ces fichiers sources doivent être suivis d'une extension quelconque : NOM. EXT, ou NOM a au plus six lettres et EXT trois.

Ceci permet lors du codage d'attribuer le même nom au fichier code, mais en supprimant l'extension.

Codage.

Les programmes CODE1 et CODE2 permettent de coder des tableaux sources de variables logiques de type 1, respectivement de type 2.

Conventions de codage communes à CODE1 et CODE2 :

- chaque symbole significatif (CONTROLE-I !) occupe 2 bytes en mémoire,
- le CONTROLE-I (tabulateur) ainsi que le LINE-FEED sont ignorés,
- le SPACE séparant variables d'entrée et de sortie est CODE 7F20 (HEX) (20H = SPACE en ASCII),
- le CARRIAGE-RETURN est codé 7F0D (HEX) (0DH = CR en ASCII),
- le "VIDE", qui apparaît généralement en cours de processus, et qui sert à indiquer qu'une variable est destinée à être supprimée, est codé 7FFF (HEX).
- On peut remarquer que le 7FH sert à distinguer les variables logiques des symboles spéciaux, ce qui explique le nombre 32511 (7EFFH = 7F00H-1) limitant le nombre de variables de type 2.

Conventions de codage particulières à CODE1:

- le "LOWEST" byte contient le CODE ASCII de l'indice,
- le "HIGHEST" byte contient le CODE ASCII de la lettre,
- cependant, le "HIGHEST" bit qui correspond habituellement au bit de signe pour les nombres, est affecté de la valeur 1 ou 0 suivant que la variable est niée ou non (le CODE ASCII n'utilise que 7 bits !).

Exemples :

		LOW HIGH BYTE
a	est codé	30H 41H
$\bar{a}$	est codé	30H 01H
a_0	est codé	30H 41H
$\bar{a}_0$	est codé	30H 01H
a_1	est codé	31H 41H
$\bar{a}_1$	est codé	31H 01H
b_4	est codé	34H 42H
$\bar{b}_4$	est codé	34H 02H

Conventions de codage particulières à CODE2 :

- La lettre est ignorée, ne jouant qu'un rôle de séparateur.
- L'indice, variant de 0 à 32511, est stocké comme un nombre habituel (par défaut 0).
- Semblablement à CODE1, le "highest" bit sert à indiquer si la variable est niée ou non (1 ou 0).

Exemples :

		Low byte	High byte
a	est codé	0000 0000	0000 0000
b	est codé	0000 0000	0000 0000
X	est codé	0000 0000	0000 0000
$\bar{X}$	est codé	0000 0000	1000 0000
X_0	est codé	0000 0000	0000 0000
$\bar{X}_0$	est codé	0000 0000	1000 0000
X_1	est codé	0000 0001	0000 0000
$\bar{X}_1$	est codé	0000 0001	1000 0000
X_2	est codé	0000 0002	0000 0000
$\bar{X}_2$	est codé	0000 0002	1000 0000
X_{32511}	est codé	1111 1111	0111 1110
$\bar{X}_{32511}$	est codé	1111 1111	1111 1110

Remarque : Codage du 1 et du 0 logique :

Le 0 logique est implicite : c'est un tableau nul, c'est-à-dire sans ligne, de longueur 0 byte.

Le 1 logique est aussi implicite : c'est une ligne sans variables; il n'est donc composé suivant les conventions de codage que d'un "space" (207F HEX) et d'un "CR" (0D7F HEX).

Ce fichier appelé UNIT dans les exemples apparaît comme suit :

UNIT :

1

Remarque importante :

Aussi bien dans les fichiers sources que tout au long du traitement, les variables d'un produit logique sont toujours en ordre croissant.

A savoir :

Type 1 : $a, a_1, a_2, \ldots, a_9, b, b_1, b_2, \ldots, b_9, \ldots, z, z_1, z_2, \ldots, z_9$

Type 2 : $1, l_1, l_2, l_3, l_4, \ldots, l_{32310}, l_{32311}$

Ceci permet de restreindre notoirement le temps de travail en linéarisant le traitement.

En effet, toute recherche d'une variable quelconque en vue par exemple d'une comparaison s'arrête dès que l'on rencontre une variable plus grande ou égale.

Exemple :

Soit deux produits :

$P_1 : a_1 a_2 \ldots a_N$

$P_2 : b_1 b_2 \ldots b_M$

On doit effectuer le produit logique $P_1 \cdot P_2$ selon les règles de l'algèbre de Boole (en particulier :

$v_1 \cdot v_2 = 0$ si $v_1 = \bar{v}_2$

$v_1 \cdot v_2 = v_1$ si $v_1 = v_2$

$v_1 \cdot v_2 = 0$ si $v_1 = 0$ ou $v_2 = 0$)

Il est donc nécessaire, si les variables ne sont pas ordonnées, d'effectuer N · M comparaisons

(a_1 doit être comparé à $b_1, b_2, \ldots, b_M$,
a_2 doit être comparé à $b_1, b_2, \ldots, b_M$,
...,
a_N doit être comparé à $b_1, b_2, \ldots, b_M$).

En revanche, si les variables sont en ordre croissant, le nombre de comparaisons ne s'élève qu'à N + M :

$a_I < a_{I+1}$, $b_J < b_{J+1}$

alors grâce à un double système de pointeurs Pt_1 et Pt_2 :

$Pt_1 = 1$ (Pt_1 pointe sur a_1)

$a_1 a_2, \ldots a_N$

situation au départ

$b_1 b_2, \ldots b_M$

$Pt_2 = 1$ (Pt_2 pointe sur b_1)

si $a_I = b_J$, on insère a_I et on fait $Pt_1 = Pt_1 + 1$, $Pt_2 = Pt_2 + 1$

Si $a_I < b_J$, on insère a_I et on fait $Pt_1 = Pt_1 + 1$;

Si $a_I > b_J$, on insère b_J et on fait $Pt_2 = Pt_2 + 1$;

Dès qu'un des produits est épuisé, on insère les variables de l'autre produit et l'on s'arrête.

Il est clair que si $a_I = \bar{b}_J$, le résultat vaut 0 et l'on peut s'arrêter immédiatement (c.f. cependant l'organigramme de la procédure PRODN).

Dans le cas normal, Pt_1 varie donc de 1 à N et Pt_2 de 1 à M, et l'on effectue ainsi au plus N + M comparaisons.

Cette propriété de l'ordre est aussi exploitée dans les tests d'absorbtion, qui représentent la majeure partie du temps d'exécution.

Description des programmes principaux.

CODE1

Appel : CODE1 FICHIER

Ce programme code suivant les conventions relatives aux variables de type 1 un fichier dont le nom est suivi d'une extension quelconque (NOM. EXT).

Le fichier produit a le même nom que le fichier source, mais sans l'extention (FICHIER. EXT FICHIER).

CODE2

Appel : CODE2 FICHIER

Ce programme effectue le même travail que CODE1, pour les variables de type 2.

OUT1

Appel : OUT1 FICHIER : LP :

Ce programme permet de lister le contenu d'un fichier code, contenant des variables de type 1.

Si : LP : est omis, l'output est listé sur la console au lieu de l'imprimante (LINE PRINTER).

OUT2

Appel : OUT2 FICHIER : LP :

Ce programme effectue le même travail que OUT1 pour les variables de type 2.

INDEX

Appel : INDEX FICHIER

Ce programme permet d'indicer un fichier contenant des variables logiques de type 1.

A chaque ligne est ajoutée une variable d'indice, qui est pourtant traitée exactement comme une quelconque variable logique.

Cet indice varie de 0_0 à 9_z de la manière suivante :

00,01,02,...,09,0a,0b,...,0z

10,11,12,...,19,1a,1b,...,1z,

....

90,91,92,...,99,9a,9b,...,9z

Ce programme permet à SHARP1 de calculer les couvertures du système.

ABDEPP

Appel : ABDEPP FICHIER

Ce programme permet d'éliminer toutes les lignes redondantes d'un fichier, c'est-à-dire de supprimer les lignes représentant des impliquants non premiers.

Le programme utilise la procédure ABDEPP qui teste l'absorbtion entre chaque ligne du fichier et le reste du fichier.

La procédure COMPAC compactifie le fichier où se trouvent des lignes codées par la procédure DEL pour signifier qu'elles doivent être supprimées.

NEGALL, NEGIN, NEGOUT

Appel : NEG? FICHIER1 FICHIER2 (? = ALL ou IN ou OUT)

Ces trois programmes permettent de nier les variables d'un FICHIER1, le résultat étant rangé dans FICHIER2.

NEGALL nie toutes les variables, NEGIN seulement les variables d'entrée et NEGOUT seulement celles de sortie.

SHARP

Appel : SHARP FICHIER1 FICHIER2 FICHIER3

Ce programme réalise l'opérateur booléen FICHIER1 • $\overline{\text{FICHIER2}}$.

Le résultat est rangé dans FICHIER3.

Le programme SHARP, qui fournit par exemple les impliquants premiers d'un système, a été modifié pour former plusieurs variantes (SHARP0, SHARP1, SHARP2).

Il a été aussi transformé en procédure (sous-routine), sous le nom de PSHARP, et est ainsi utilisé par le programme simpl.

Cet opérateur SHARP permet d'effectuer toutes les opérations logiques habituelles (on peut le comparer par analogie aux portes NAND ou NOR).

La description des opérations effectuées dans ce programme est la suivante :

TABLEAUX DE DEPART

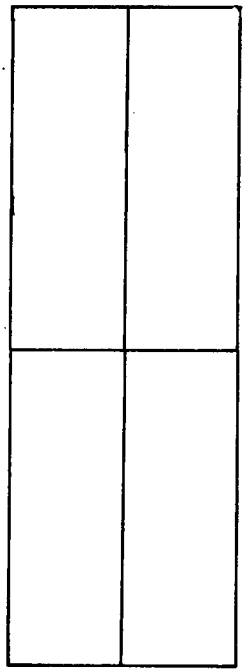
V	V
A	A
R	R
D	D
'	E
E	
N	S
T	O
R	R
E	T
E	I
	E

T_1

V	V
A	A
R	R
D	D
'	E
E	
N	S
T	O
R	R
E	T
E	I
	E

T_2

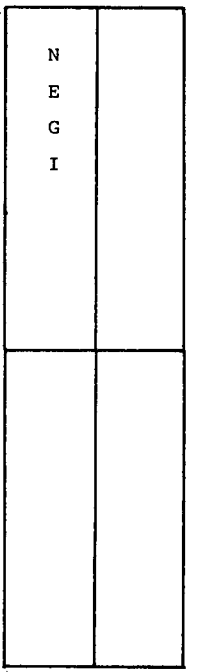
SITUATION DE DEPART



T_2

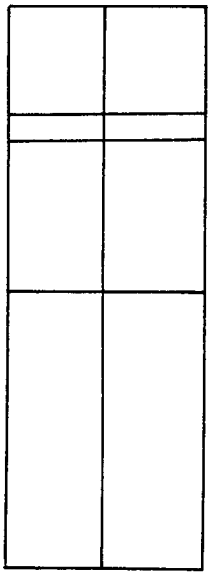
T_1

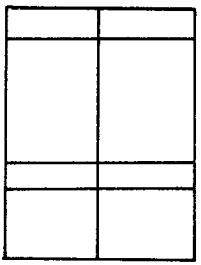
PREMIERE PHASE



On nie les variables d'entrée de T_2

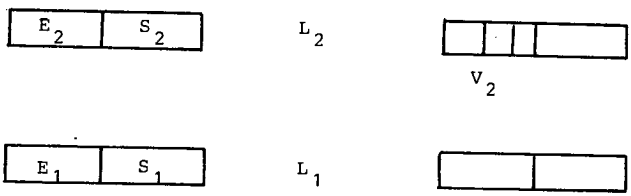
DEUXIEME PHASE



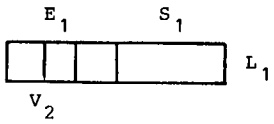


On itère sur chaque ligne L_2 de T_2, avec chaque ligne L_1 de T_1, la phase 3.

TROISIEME PHASE



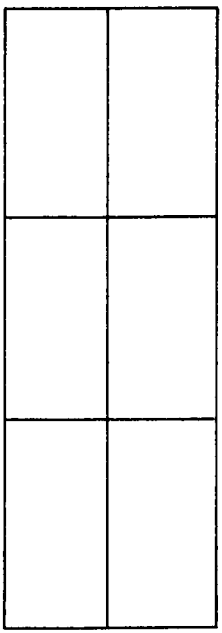
On itère sur chaque variable d'entrée V_2 de L_2 l'opération suivante :



On insère V_2 dans le produit E_1, constitué des variables d'entrée de la ligne L_1, d'après les règles de l'algèbre de Boole.

Au cours de la troisième phase, on crée donc ligne après ligne le tableau intermédiaire T_3.

A la fin de la phase 3, les tableaux T_1 et T_3 sont réunis en un seul, sous le nom de F_1.

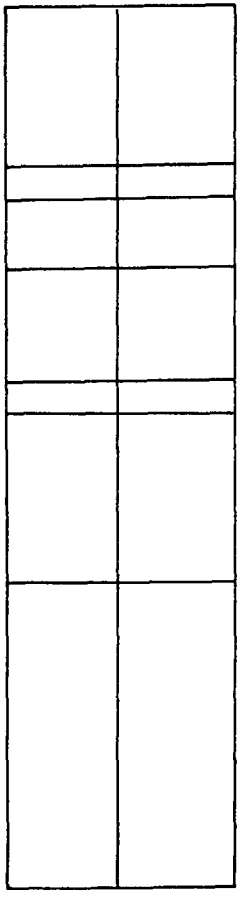


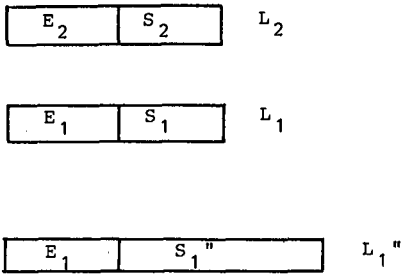
Note : La phase 3 peut être inhibée par la fonction sinclu si une variable de L_2 est identique à une variable de L_1.
En effet, les tests d'absorbtion décrits plus loin supprimeraient toute ligne crée dans de telles conditions.

$T_1 = T_1 + T_3$

QUATRIEME PHASE

A la fin de chaque troisième phase, on effectue l'opération suivante :





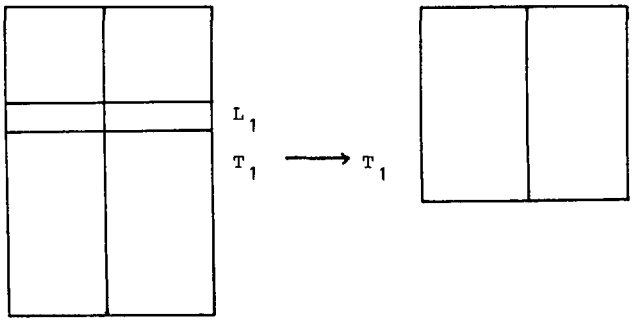
Les variables de sortie de L_1'' sont formées par le produit de S_1 et S_2 suivant les règles de l'algèbre de Boole.

Remarque : si une variable apparaît à la fois sous sa forme positive et sous sa forme niée, cette dernière seule est insérée.

A la fin de chaque quatrième phase, on effectue les tests d'absorbtion suivants pour simplifier au maximum les tableaux T_1 et T_2 :

CINQUIEME PHASE

TEST ADL



La ligne courante de T_1 est comparée avec toutes les autres lignes de ce tableau.
Si une ligne de T_1 absorbe L_1, ou le contraire, la ligne idoine est supprimée.

Absorbtion :

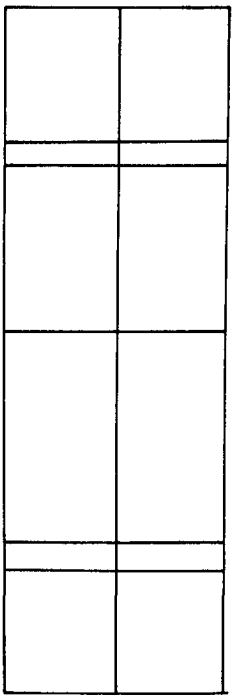
ab f	absorbe	abc f
ab f	"	ab fg
ab f	"	abc fg

etc...

En termes simples, absorbtion signifie que toutes les variables du produit 1 figurent dans le produit 2, alors 1 absorbe 2. La comparaison est faite indépendamment pour les variables de sortie et celles d'entrée: il faut le consensus entre ces 2 tests pour qu'il y ait absorbtion.

SIXIEME PHASE

Test ADC



Chaque ligne de T_1 est comparée à chaque ligne de T_3.

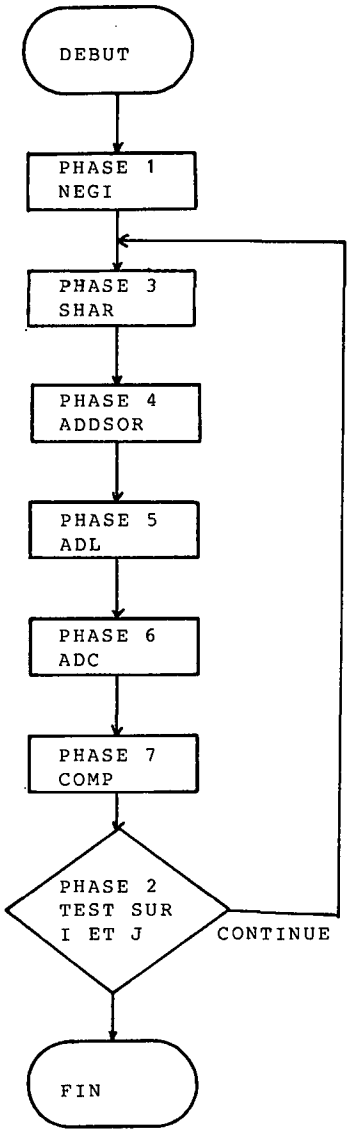
On supprime les lignes qui sont absorbées.

A la fin de ADL et ADC, les tableaux T_1 et T_3 sont compactifiés et réunis en un : T_1.

(PHASE 7 : comp)

Correspondance entre phases et noms de sous-routines.

L'organigramme qui décrit sommairement le programme SHARP par la même occasion est le suivant :



SHARP0

Appel : SHARP0 FICHIER1 FICHIER2 FICHIER3

Cette variante du programme SHARP permet de calculer toutes les couvertures partielles d'un système, le résultat étant rangé dans FICHIER3 (c.f. les exemples pour la marche à suivre).

SHARP1

Appel : SHARP1 FICHIER1 FICHIER2 FICHIER3

Ce programme est semblable au programme SHARP0, mais ne fournit que les couvertures complètes du système.

SHARP2

Appel : SHARP2 FICHIER1 FICHIER2 FICHIER3 N

(par défaut, N prend la valeur 1).

Ce programme permet de traiter de manière heuristique le problème des couvertures.

Comme des calculs de couvertures ont une complexité principalement liée au résultat, il ne faut donc pas se fier à l'apparente simplicité du tableau de départ.

Le programme SHARP2 permet de ne produire que les N premières lignes du résultat intermédiaire (tableau T_3 de la troisième phase de la description du programme SHARP).

Les couvertures ainsi trouvées ne seront pas nécessairement :

- premièrement, minimales, mais, comme on peut le constater en pratique, sont relativement proches des solutions minimales complètes;
- deuxièmement, irrédondantes, mais, pratiquement, proches des solutions irrédondantes données par SHARP0 par exemple.

SIMPL

Appel : SIMPL FICHIER1 FICHIER2

Ce programme s'inspire aussi d'une approche heuristique.

Il effectue l'opérateur SHARP entre chaque ligne de FICHIER1 et le reste de ce tableau. Si le résultat est nul, la ligne courante est supprimée, sinon elle est conservée.

Le résultat est rangé dans FICHIER2.

Introduction aux organigrammes.

Pour simplifier la présentation des organigrammes, les conventions suivantes sont utiles :

DE_1 : adresse des variables d'entrée de la ligne 1
DE_2 : adresse des variables d'entrée de la ligne 2
DS_1 : adresse des variables de sortie de la ligne 1
DS_2 : adresse des variables de sortie de la ligne 2

E_1 : variable d'entrée courante de la ligne 1
E_2 : variable d'entrée courante de la ligne 2
S_1 : variable de sortie courante de la ligne 1
S_2 : variable de sortie courante de la ligne 2

Pour signifier que l'on passe de la variable courante V à la suivante, on écrit : NEXT (V).

Pour indiquer que l'on commence par la première variable d'un produit, on écrit V = 1.

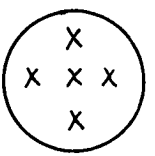
Pour indiquer qu'une variable est sous une forme négative, niée ($\bar{V}$), on pose NEG(V) = 1, sinon NEG(V) = 0.

Pour indiquer que l'on est à la fin d'un produit de variables, on pose : V = CODE, sinon V <> CODE.

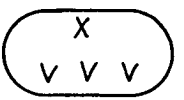
Parfois, on écrit : I = FONCTION (V_1,V_2). Se référer à l'organigramme décrivant cette fonction.

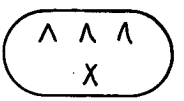
Pour indiquer qu'un organigramme est décomposé en 2 parties, on utilise les symboles suivants :

Organigramme principal :



Début et fin de la partie reportée :





Remarque : l'organigramme de la procédure ABSONG est identique à celui de la procédure ABSO, mis à part que ABSONG ne considère que les variables sous forme niée.

Organigramme de la fonction ABSORB.

Cette fonction teste l'absorbtion entre deux lignes. Pour les variables de sortie, seules les variables négatives sont prises en compte.

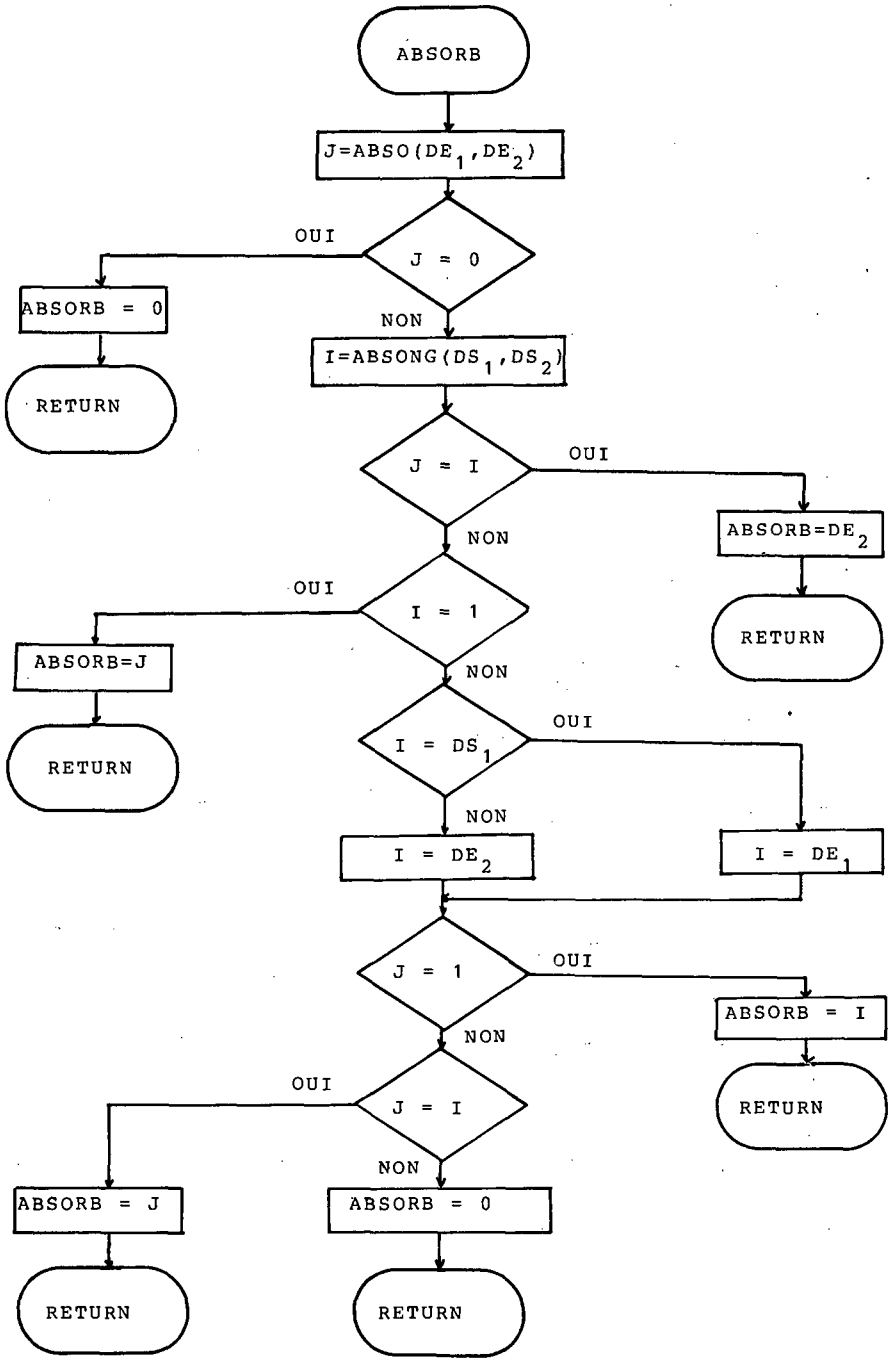
ABSORB(DE_1,DE_2) prend la valeur :

1 si les deux lignes sont identiques

0 si il n'y a pas d'absorbtion

DE_1 si $E_1 < E_2$ et $S_1 < S_2$

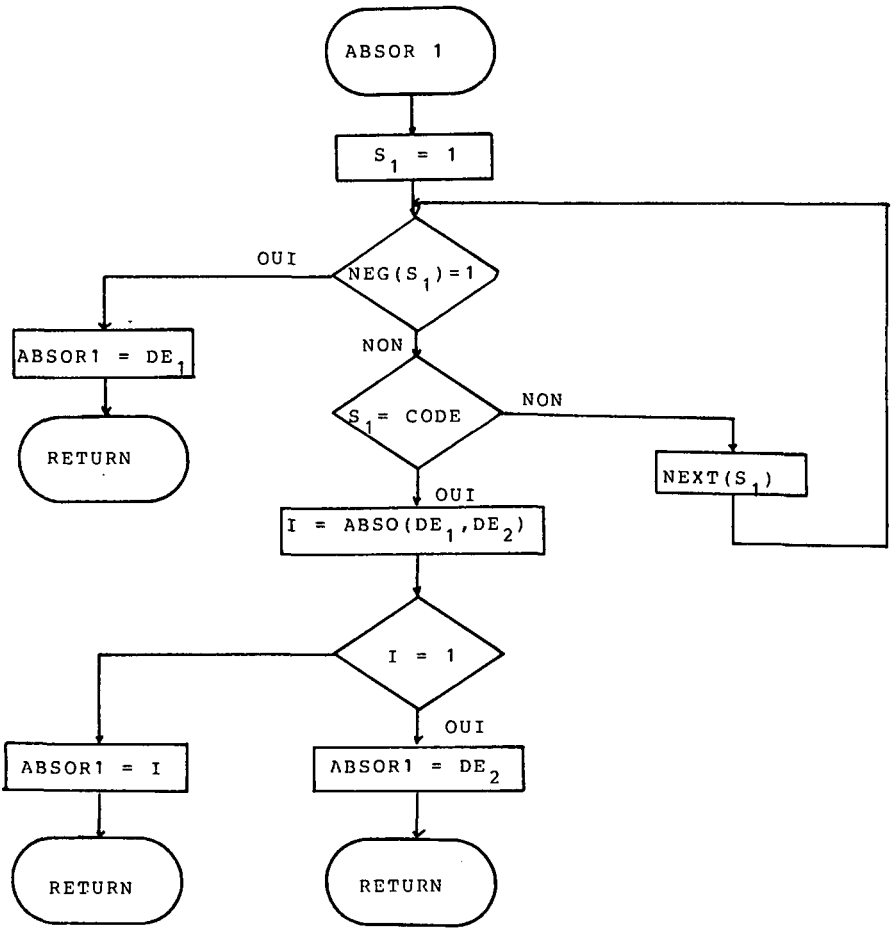
DE_2 si $E_1 > E_2$ et $S_1 > S_2$


ABSORB
J=ABSO(DE1,DE2)
J = 0
OUI
ABSORB = 0
RETURN
NON
I=ABSONG(DS1,DS2)
J = I
OUI
ABSORB=DE2
RETURN
NON
I = 1
OUI
ABSORB=J
RETURN
NON
I = DS1
OUI
I = DE1
NON
I = DE2
J = 1
OUI
ABSORB = I
RETURN
NON
J = I
OUI
ABSORB = J
RETURN
NON
ABSORB = 0
RETURN

Organigramme de la fonction ABSORB1.

Cette fonction est semblable à la fonction ABSORB, mais elle provoque la suppression des lignes ayant des variables de sortie négative.

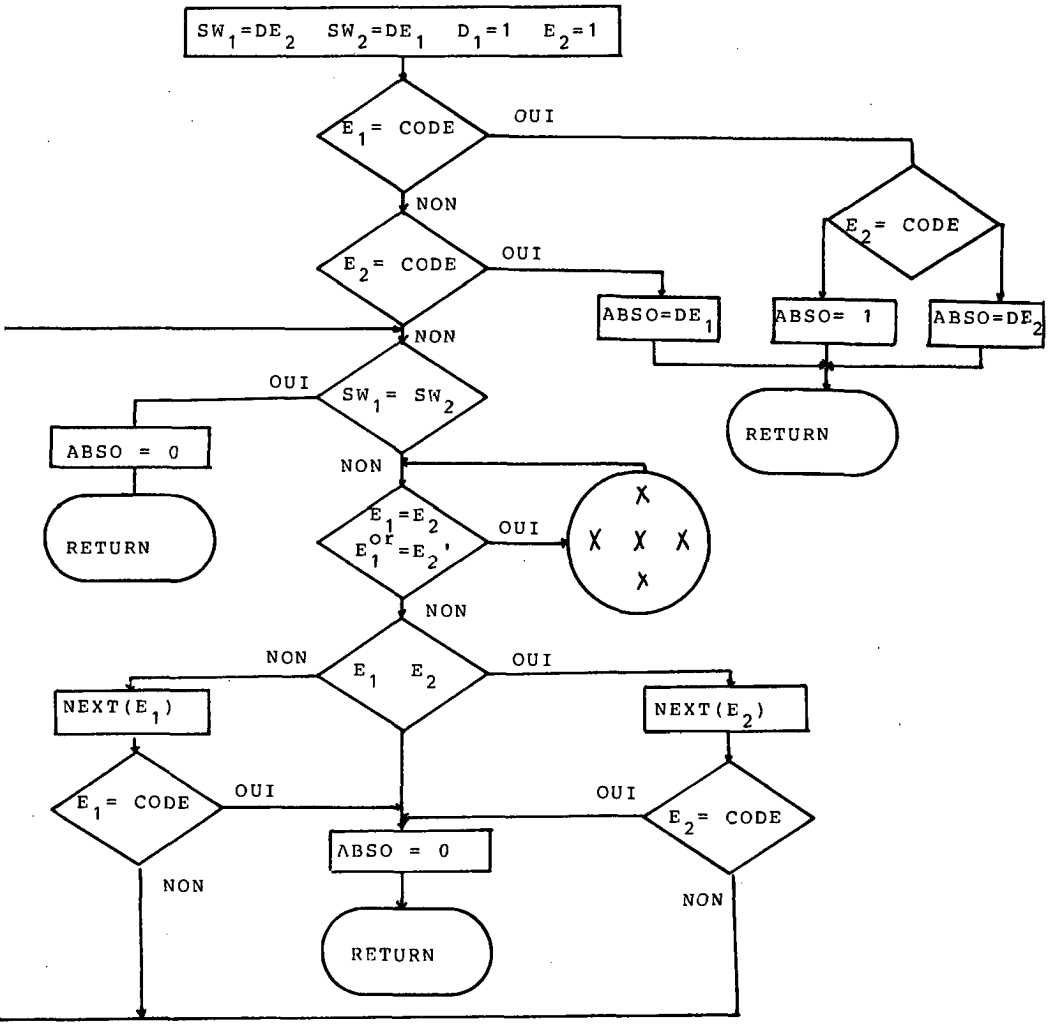
Comme un fichier résultat n'a aucune variable de sortie négative avec cette procédure, celles-ci ne peuvent apparaître que en raison d'ABSORB (où on ajoute les variables de sortie de la source à la ligne courante). Par conséquent, le test de "négativité" n'est fait que sur un des produits de variable, à savoir S_1.

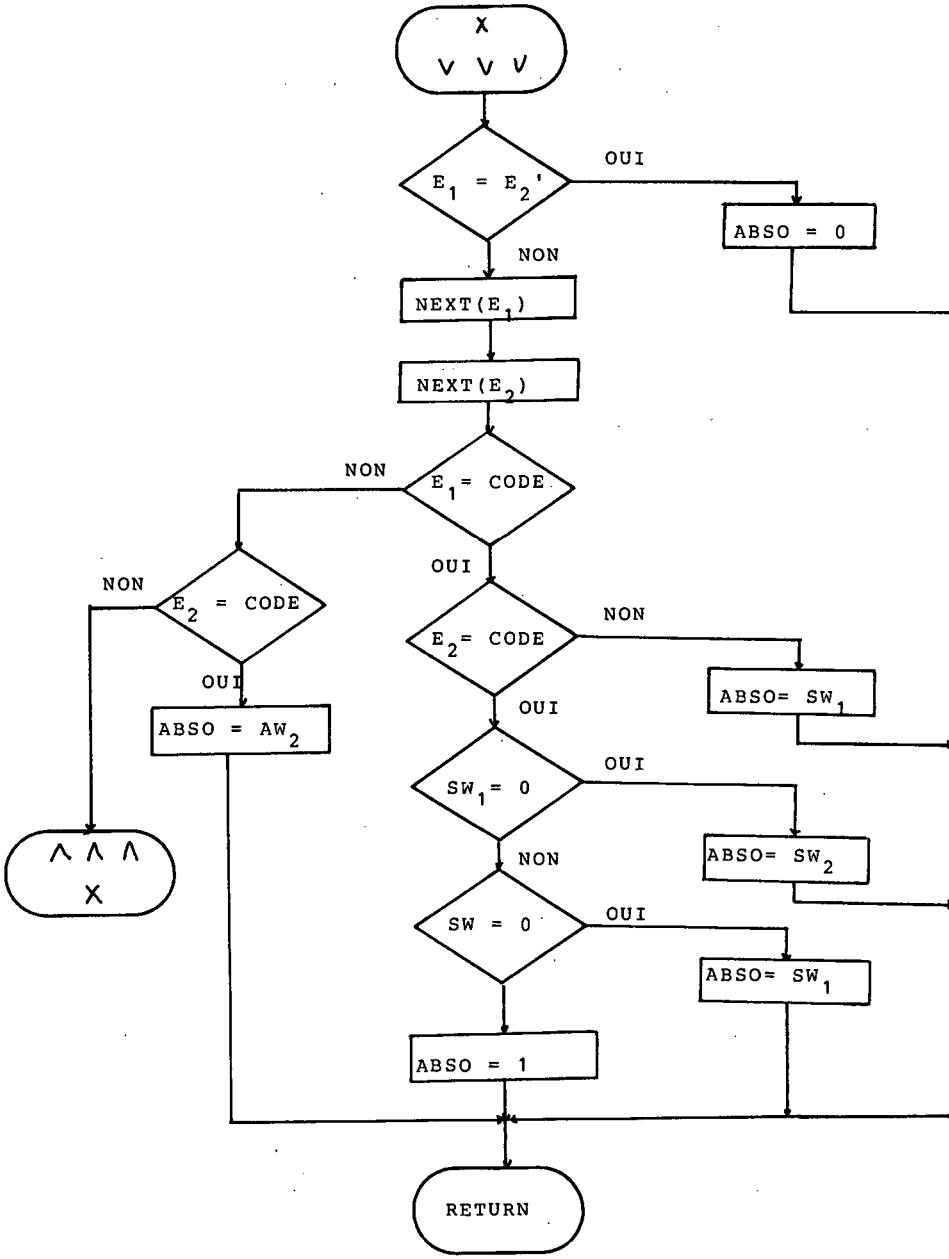


Organigramme de la fonction ABSO.

Cette fonction teste l'absorbtion entre toutes les variables de deux produits. (Variables d'entrée).
La fonction ABSONG est identique, mais ne considère que les variables négatives. (Variables de sortie).

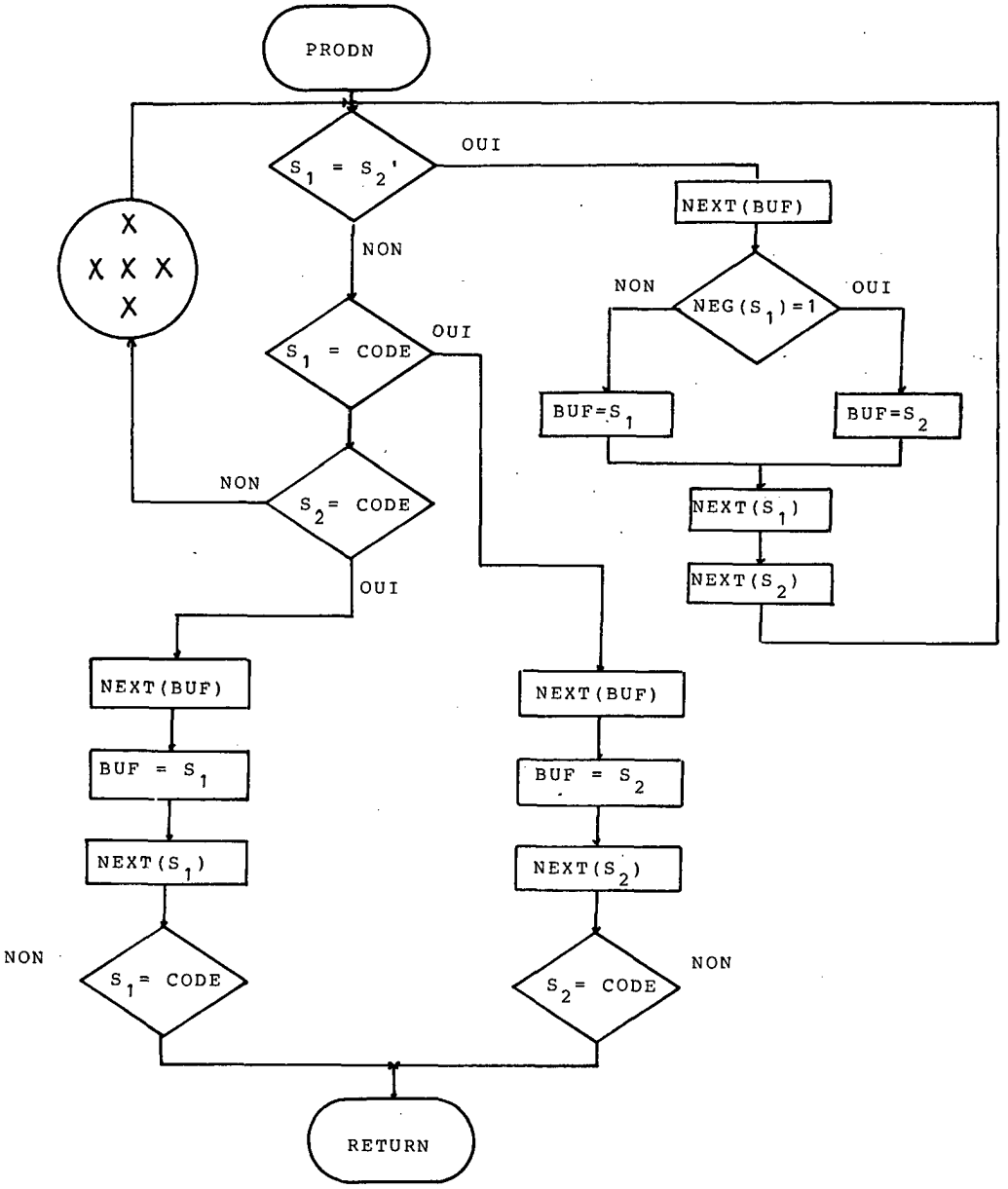
ABSO (DE_1,DE_2) prend la valeur 1 si $E_1 = E_2$
0 s'il n'y a pas d'absorbtion
DE_1 si $E_1 \leqslant E_2$
DE_2 si $E_1 \geqslant E_2$

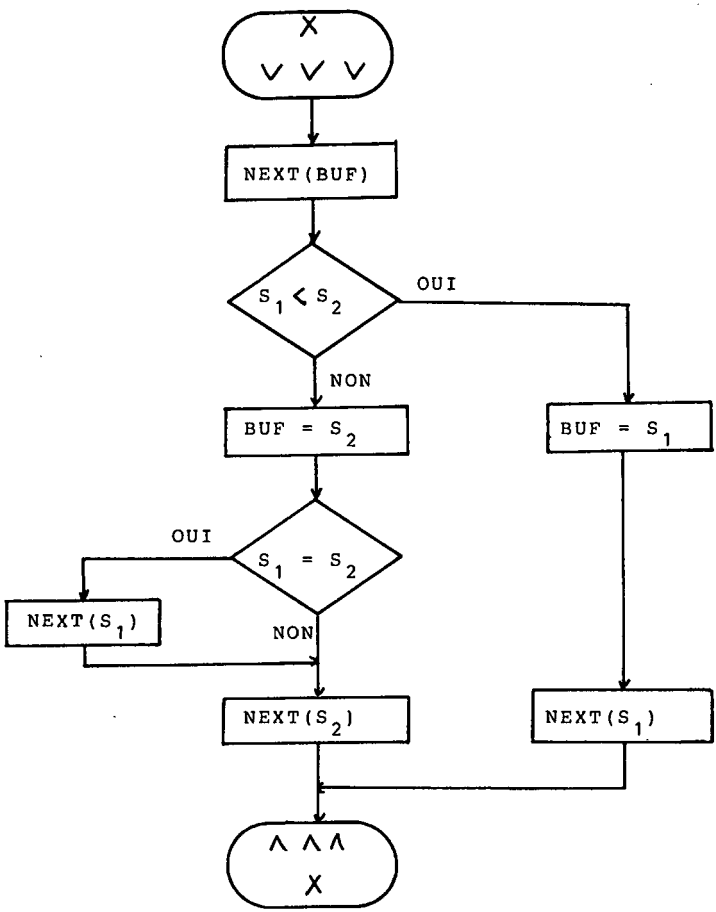



X
V V V
E1 = E2'
OUI
ABSO = 0
NON
NEXT(E1)
NEXT(E2)
E1 = CODE
NON
OUI
E2 = CODE
NON
OUI
ABSO = AW2
E2 = CODE
NON
ABSO= SW1
OUI
SW1 = 0
OUI
ABSO= SW2
NON
SW = 0
OUI
ABSO= SW1
ABSO = 1
RETURN
^ ^ ^
X

Organigramme de la procédure PRODN.

Cette procédure réalise le produit logique de deux produits de variables, suivant les règles de l'algèbre de Boole.





Remarque : la procédure PRODN présente la particularité de conserver pour le résultat une variable sous sa forme niée si l'on est en présence de cette variable à la fois sous sa forme positive et sous sa forme niée.

ANNEXE 2.

Exemple de la simplification d'une PLA industrielle.

Cette PLA d'un circuit réalisé pour une montre électronique apparaît comme suit :

1	A B C $\bar{D}$ E $\bar{F}$ $\bar{G}$ $\bar{H}$	K L M $\bar{N}$ O
2	A B C $\bar{D}$ $\bar{E}$ $\bar{F}$ $\bar{H}$	K L $\bar{M}$ N O
3	A B $\bar{C}$ D E $\bar{F}$ $\bar{H}$	K $\bar{L}$ M N O
4	A B $\bar{C}$ $\bar{D}$ E $\bar{F}$ G $\bar{H}$ $\bar{J}$	K L M $\bar{N}$ O
5	A B $\bar{C}$ D $\bar{E}$ $\bar{F}$ G $\bar{H}$ J	K L M $\bar{N}$ O
6	A B $\bar{C}$ $\bar{D}$ $\bar{E}$ $\bar{F}$ $\bar{G}$ H	K $\bar{L}$ M N O
7	A B $\bar{C}$ D $\bar{E}$ $\bar{F}$ $\bar{G}$ H	K L $\bar{M}$ N O
8	A B $\bar{C}$ $\bar{D}$ E $\bar{F}$ $\bar{G}$ H	K L M $\bar{N}$ O
9	A $\bar{B}$ $\bar{C}$ $\bar{D}$ E $\bar{F}$ G H	K L $\bar{M}$ N O
10	A $\bar{B}$ $\bar{C}$ $\bar{D}$ $\bar{E}$ $\bar{F}$ G H	K $\bar{L}$ M N O
11	A $\bar{B}$ $\bar{C}$ D E $\bar{F}$ G H	$\bar{K}$ L M $\bar{N}$ O
12	$\bar{A}$ B $\bar{C}$ D $\bar{E}$ F $\bar{G}$ $\bar{H}$	K L M $\bar{N}$ O
13	$\bar{A}$ B $\bar{C}$ $\bar{D}$ E F $\bar{G}$ $\bar{H}$	K L $\bar{M}$ N O
14	$\bar{A}$ B $\bar{C}$ $\bar{D}$ $\bar{E}$ F $\bar{G}$ $\bar{H}$	K $\bar{L}$ M N O
15	$\bar{A}$ $\bar{B}$ C D E F $\bar{G}$ H	K L M $\bar{N}$ O
16	$\bar{A}$ $\bar{B}$ C D $\bar{E}$ F $\bar{G}$ H	K L $\bar{M}$ N O
17	$\bar{A}$ $\bar{B}$ C $\bar{D}$ E F $\bar{G}$ H	K $\bar{L}$ M N O
18	$\bar{A}$ $\bar{B}$ C D E F G H I	K L M $\bar{N}$ O
19	$\bar{A}$ $\bar{B}$ C D $\bar{E}$ F G H I	K L $\bar{M}$ N O
20	$\bar{A}$ $\bar{B}$ C $\bar{D}$ E F G H I	K $\bar{L}$ M N O

21	$\bar{A}\ \bar{B}\ C\ D\ \bar{E}\ F\ G\ H\ \bar{I}\ \bar{J}$	$K\ L\ M\ \bar{N}\ O$
22	$\bar{A}\ \bar{B}\ C\ D\ \bar{E}\ F\ G\ H\ \bar{I}\ \bar{J}$	$K\ L\ \bar{M}\ N\ O$
23	$\bar{A}\ \bar{B}\ C\ \bar{D}\ E\ F\ G\ H\ \bar{I}\ \bar{J}$	$K\ \bar{L}\ M\ N\ O$
24	$\bar{A}\ \bar{B}\ C\ \bar{D}\ \bar{E}\ F\ G\ H\ \bar{I}\ J$	$K\ \bar{L}\ M\ N\ O$
25	$\bar{A}\ \bar{B}\ \bar{C}\ D\ E\ F\ G\ \bar{H}$	$K\ L\ M\ \bar{N}\ O$
26	$\bar{A}\ \bar{B}\ \bar{C}\ D\ \bar{E}\ F\ G\ \bar{H}$	$K\ L\ \bar{M}\ N\ O$
27	$\bar{A}\ \bar{B}\ \bar{C}\ \bar{D}\ E\ F\ G\ \bar{H}$	$K\ \bar{L}\ M\ N\ O$
28	$\bar{A}\ \bar{B}\ C\ \bar{D}\ E\ F\ G\ H\ \bar{I}\ J$	$\bar{K}\ L\ \bar{M}\ N\ O$
29	$\bar{A}\ \bar{B}\ C\ D\ \bar{E}\ F\ G\ H\ \bar{I}\ J$	$\bar{K}\ L\ M\ \bar{N}\ O$
30	$A\ B\ C\ D\ \bar{E}$	$K\ L\ M\ N\ \bar{O}$

Le résultat de la simplification donne :

1	$\bar{D}\ E\ H$	$\bar{K}\ L\ \bar{M}\ \bar{N}\ \bar{O}$
2	$\bar{D}\ E\ \bar{F}\ \bar{H}$	$\bar{K}\ \bar{L}\ \bar{M}\ N\ \bar{O}$
3	$\bar{D}\ \bar{E}\ F$	$\bar{K}\ L\ \bar{M}\ \bar{N}\ \bar{O}$
4	$C\ D\ \bar{H}$	$\bar{K}\ \bar{L}\ \bar{M}\ \bar{N}\ O$
5	$C\ \bar{F}\ H$	$\bar{K}\ \bar{L}\ \bar{M}\ \bar{N}\ O$
6	$B\ F\ H$	$\bar{K}\ \bar{L}\ \bar{M}\ \bar{N}\ O$
7	$D\ E\ F$	$\bar{K}\ \bar{L}\ \bar{M}\ N\ \bar{O}$
8	$E\ \bar{F}\ \bar{G}\ H$	$\bar{K}\ \bar{L}\ \bar{M}\ N\ \bar{O}$
9	$D\ E\ \bar{F}\ \bar{H}$	$\bar{K}\ L\ \bar{M}\ \bar{N}\ \bar{O}$
10	$\bar{B}\ \bar{C}\ D\ H$	$K\ \bar{L}\ \bar{M}\ N\ \bar{O}$
11	$E\ F\ \bar{G}\ \bar{H}$	$\bar{K}\ \bar{L}\ M\ \bar{N}\ \bar{O}$
12	$E\ F\ G\ H\ \bar{I}\ J$	$K\ \bar{L}\ M\ \bar{N}\ \bar{O}$
13	$\bar{B}\ \bar{C}\ \bar{D}\ E\ H$	$\bar{K}\ \bar{L}\ M\ \bar{N}\ \bar{O}$
14	$\bar{D}\ F\ \bar{G}\ H$	$\bar{K}\ L\ \bar{M}\ \bar{N}\ \bar{O}$
15	$\bar{D}\ F\ G\ \bar{H}$	$\bar{K}\ L\ \bar{M}\ \bar{N}\ \bar{O}$

16	$\bar{D}$ F G I	$\bar{K}$ L $\bar{M}$ $\bar{N}$ $\bar{O}$
17	$\bar{D}$ F G $\bar{J}$	$\bar{K}$ L $\bar{M}$ $\bar{N}$ $\bar{O}$
18	$\bar{B}$ $\bar{C}$ D $\bar{E}$	$\bar{K}$ $\bar{L}$ M $\bar{N}$ $\bar{O}$
19	$\bar{C}$ D $\bar{E}$ $\bar{F}$ $\bar{G}$	$\bar{K}$ $\bar{L}$ M $\bar{N}$ $\bar{O}$
20	$\bar{B}$ C D G $\bar{I}$ J	K $\bar{L}$ $\bar{M}$ N $\bar{O}$
21	$\bar{A}$ C $\bar{E}$ $\bar{G}$	$\bar{K}$ $\bar{L}$ M $\bar{N}$ $\bar{O}$
22	C $\bar{D}$ $\bar{E}$ $\bar{H}$	$\bar{K}$ $\bar{L}$ M $\bar{N}$ $\bar{O}$
23	$\bar{B}$ C $\bar{E}$ I	$\bar{K}$ $\bar{L}$ M $\bar{N}$ $\bar{O}$
24	$\bar{B}$ C $\bar{E}$ $\bar{J}$	$\bar{K}$ $\bar{L}$ M $\bar{N}$ $\bar{O}$
25	B $\bar{C}$ D $\bar{E}$ $\bar{H}$	$\bar{K}$ $\bar{L}$ $\bar{M}$ N $\bar{O}$

On constate que cette PLA, une fois simplifiée, ne contient plus que 25 lignes au lieu de 30, ce qui représente un gain en surface de 17 %. Le nombre de transistors nécessaire, c'est-à-dire, le nombre de lettres de chaque produit, diminue du plus de 50 %.