Minimisation de fonction par software (1),(2)

17 octobre 2017

Introduction

Problématique

Comment minimiser une fonction à 10 , 20 ou 50 variables ?

Solution

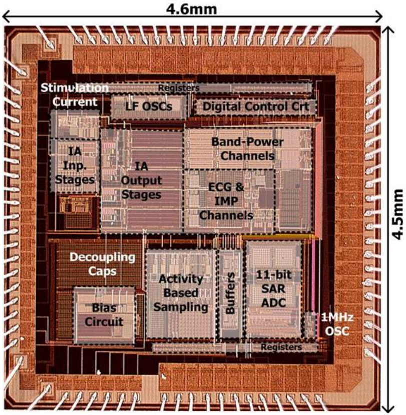
L'algorithme Espresso développé par IBM (3) (1984) et amélioré par l'université de Berkley (4) (1986)

Utilisation

Dans les logiciels de conception de ASIC (Application-Specific Integrated Circuit) ou FPGA (Field-Programmable Gate Array)

Introduction

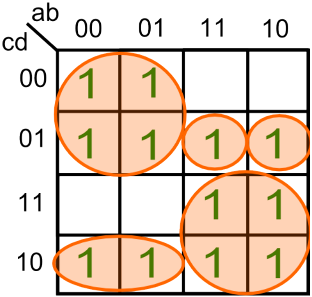
Exemple de ASIC, un processeur [1] pour traitement du signal développé par Imec .



1. Imec and Holst Centre's 30μW analog signal processor ASIC for biopotential signal monitoring, 2010, available at http://start.artoos.eu/dmp/printflo/content/107/96/forward.aspx?id=176395

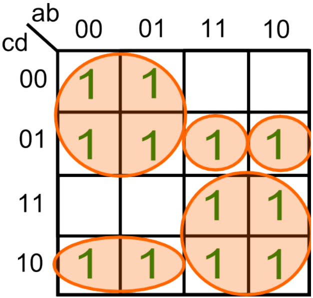
La boucle Reduction-Expansion-Irredondance

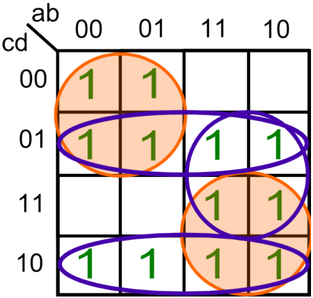
Partons d'un diagramme de Karnaugh représentant une fonction non-minimale à 5 impliquants.



La boucle Reduction-Expansion-Irredondance

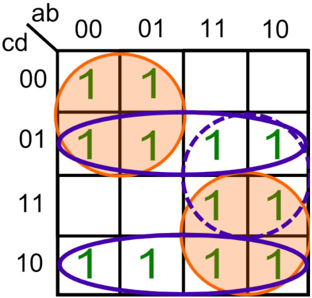
Expansons tous les impliquants au maximum afin de n'obtenir plus que des impliquants premiers. Ainsi, l'impliquant $ab\bar{c}d$ devient $\bar{c}d$, l'impliquant $a\bar{b}\bar{c}d$ devient ad et l'impliquant $\bar{a}c\bar{d}$ devient $c\bar{d}$.





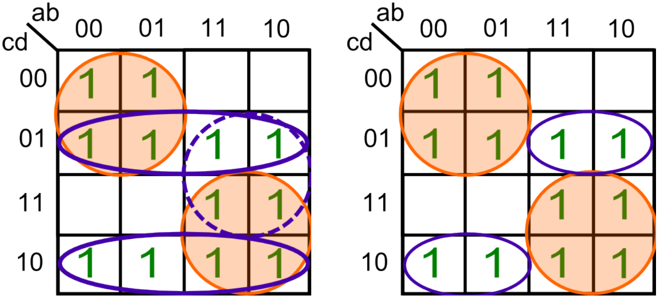
La boucle Reduction-Expansion-Irredondance

Supprimons les impliquants redondants. Les 1 de l'impliquant *ad* sont tous couverts par un autre impliquant. L'impliquant n'est donc pas utile et est redondant. On peut donc le retirer.



La boucle Reduction-Expansion-Irredondance

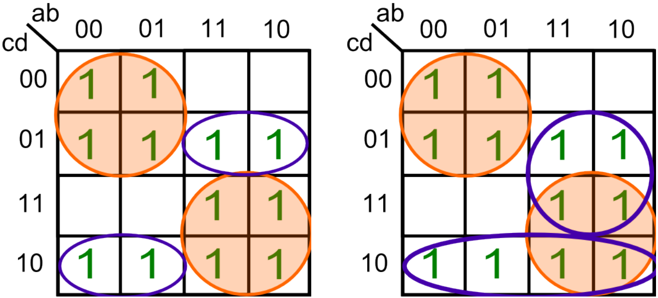
Réduisons les impliquants au minimum. Les impliquants sont réduits afin de ne couvrir que les 1 absolument indispensables, que d'autre impliquants ne couvrent pas. L'impliquant $\bar{c}d$ devient $a\bar{c}d$ et l'impliquant $c\bar{d}$ devient $\bar{a}c\bar{d}$.



La première boucle est terminée. On en commence une deuxième.

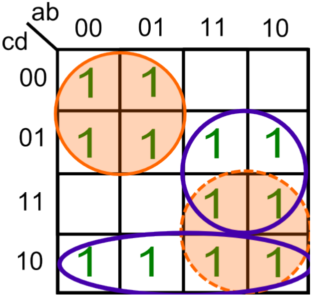
La boucle Reduction-Expansion-Irredondance

Expansons à nouveau tous les impliquants au maximum afin de n'obtenir plus que des impliquants premiers. Ainsi, l'impliquant $a\bar{c}d$ devient ad et l'impliquant $\bar{a}c\bar{d}$ devient $c\bar{d}$.



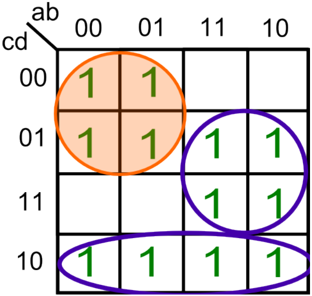
La boucle Reduction-Expansion-Irredondance

Supprimons les impliquants redondants. Nous pouvons supprimer *ac*.



La boucle Reduction-Expansion-Irredondance

Réduisons les impliquants au minimum...(il n'y a plus moyen de réduire), on y est !



Il faut généralement moins de 5 itérations pour trouver une solution minimale, irredondante, dans lequel on ne retrouve que des impliquants premiers.

Contenu

- Positional Cube Notation (PCN)
- L'expansion de Shannon
- Tautologie
- Fonction monoforme et biforme
- Unate Recursive Paradigm (URP)

Positional Cube Notation (PCN)

C'est une technique permettant de décrire informatiquement un impliquant appelé cube. Cette méthode permet d'effectuer des opérations sur des fonctions représentées par des listes d'impliquants ou cubes. Principe :

- **Pour chaque impliquant, on crée un vecteur ligne** comprenant autant de colonne qu'il y a de variable dans la fonction.

Exemple

- pour le minterm $a\bar{b}\bar{c}d$: $pcn = [.. .. .. ..]$ où chaque .. est une valeur à définir.
- pour l'impliquant $x\bar{y}$ (la fonction à laquelle appartient cet impliquant contient les variables x, y et z) : $pcn = [.. .. ..]$ où chaque .. est une valeur à définir.

Positional Cube Notation (PCN) (suite)

- **Chaque colonne représente une variable.**

Exemple

- pour le minterm $a\bar{b}\bar{c}d$:

$$\begin{array}{l} \quad\quad\; a \;\; b \;\; c \;\; d \\ \quad\quad\; \downarrow \;\; \downarrow \;\; \downarrow \;\; \downarrow \\ pcn = [\,..\;\;..\;\;..\;\;..\,] \end{array}$$

- pour l'impliquant $x\bar{y}$ (la fonction à laquelle appartient cet impliquant contient les variables x, y et z) :

$$\begin{array}{l} \quad\quad\; x \;\; y \;\; z \\ \quad\quad\; \downarrow \;\; \downarrow \;\; \downarrow \\ pcn = [\,..\;\;..\;\;..\,] \end{array}$$

Positional Cube Notation (PCN) (suite)

- **Le vecteur est rempli en regardant si chaque variable apparait, niée ou non dans l'impliquant.** Si la variable apparait :
  - non niée => écrire '01' dans la colonne de la variable
  - niée => écrire '10' dans la colonne de la variable
  - ni niée ni non niée => écrire '11' dans la colonne de la variable

Exemple

- pour le minterm $a\bar{b}\bar{c}d$: a et d apparaissent non niés (donc '01'), b et c apparaissent niés (donc '10').

$$\begin{array}{cccccc} & & a & b & c & d \\ & & \downarrow & \downarrow & \downarrow & \downarrow \\ pcn = & [& 01 & 10 & 10 & 01 \;] \end{array}$$

Positional Cube Notation (PCN) (suite)

Exemple

- pour l'impliquant $x\bar{y}$ (la fonction à laquelle appartient cet impliquant contient les variables x, y et z) : x apparait non nié ('01'), y apparait nié ('10') et z n'apparait pas ('11').

$$\begin{array}{ccccc} & x & y & z & \\ & \downarrow & \downarrow & \downarrow & \\ pcn = [& 01 & 10 & 11 &] \end{array}$$

Grâce à cette notation, nous allons pouvoir exprimer nos fonctions booléennes comme étant des listes d'impliquants ou cubes.

Exemple

Si $f = a\bar{b} + \bar{a}\bar{b}c + bc$, la liste sera : [01 10 11], [10 10 01], [11 10 10]

L'expansion de Shannon

L'expansion de Shannon permet de décomposer une fonction booléenne (comme les séries de Taylor, les séries de Fourrier).

Ce théorème fait apparaitre la notion de cofacteur, négatif ou positif.

Définition

Soit la fonction $F(x1, x2, ..., xi, ...xN)$, $F(xi = 0)$ est le cofacteur négatif de F selon xi
Soit la fonction $F(x1, x2, ..., xi, ...xN)$,$F(xi = 1)$ est le cofacteur positif de F selon xi

L'expansion de Shannon (suite)

Le cofacteur de F selon xi est simplement F pour laquelle on a remplacé xi par 1 (pour le cofacteur positif) et par 0 (pour le cofacteur négatif).

Exemple

Si $f = a\bar{b} + \bar{a}\bar{b}c + bc$,
le cofacteur positif selon a, $f(a = 1)$, vaut $\bar{b} + bc$
le cofacteur négatif selon a, $f(a = 0)$, vaut $\bar{b}c + bc$

L'expansion de Shannon (suite)

L'expansion de Shannon pose l'égalité suivante :

$$F(x1, x2, \ldots, xi, \ldots .xN) = xi * F(xi = 1) + \bar{xi} * F(xi = 0)$$

Exemple

$f = a\bar{b} + \bar{a}\bar{b}c + bc$ est il égal à $a * f(a = 1) + \bar{a} * f(a = 0)$?

$a * f(a = 1) + \bar{a} * f(a = 0)$

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

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

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

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

L'expansion de Shannon (suite)

Une fois l'expansion réalisée sur une variable, on peut le refaire pour une autre :

$$F(x1, x2, .., xi, xj, .., xN) = xi * xj * F_{xi,xj} + \overline{xi} * xj * F_{\bar{xi},xj} + xi * \overline{xj} * F_{xi,\bar{xj}} + \overline{xi} * \overline{xj} * F_{\bar{xi},\bar{xj}}$$

avec

$$F_{xi,xj} = F(xi = 1, xj = 1)$$
$$F_{\bar{xi},xj} = F(xi = 0, xj = 1)$$
$$F_{xi,\bar{xj}} = F(xi = 1, xj = 0)$$
$$F_{\bar{xi},\bar{xj}} = F(xi = 0, xj = 0)$$

Exemple

Si $F = a\overline{b} + \overline{a}\overline{b}c + bc$ alors $F_{a,\bar{c}} = \overline{b}$

L'expansion de Shannon (suite)

Comment faire pour décomposer une liste de cube en cofacteur ?

Pour obtenir **le cofacteur positif** d'une liste de cube selon une variable x, regarder dans la colonne de x de chaque cube :

- Si **'10'** => supprimer ce cube.
- Si **'01'** => remplacer ce '01' par '11'.
- Si **'11'** => ne rien faire.

Pour obtenir **le cofacteur négatif** d'une liste de cube selon une variable x, regarder dans la colonne de x de chaque cube :

- Si **'01'** => supprimer ce cube.
- Si **'10'** => remplacer ce '10' par '11'.
- Si **'11'** => ne rien faire.

Tautologie

Une fonction F est une tautologie si cette fonction vaut 1.

F est une tautologie si F == 1

On peut appliquer la théorème de Shannon pour analyser la tautologie : F sera une tautologie si ces cofacteurs positifs et négatifs selon une variable x sont aussi tous les deux une tautologie.

F est une tautologie si $F_x = 1$ ET $F_{\bar{x}} = 1$
En effet, si $F_x = 1$ et $F_{\bar{x}} = 1$, alors
$F = x * F_x + \bar{x} * F_{\bar{x}}$
$= x * 1 + \bar{x} * 1$
$= x + \bar{x}$
$= 1$

Fonction monoforme (*unate*) et biforme (*binate*)

Une fonction est dite monoforme (*unate*) si elle ne contient qu'une polarité de chaque variable. Si ce n'est pas le cas, elle est dite biforme. (*binate*).

Exemple

$$f = abd + b\bar{c}d + a\bar{c}$$

Cette fonction est **monoforme** car *a* est toujours non niée, *b* est toujours non niée, *c* est toujours niée et d est toujours non niée.

$$f = \bar{a}d + bcd + a\bar{c}$$

Cette fonction est **biforme** car *a* et *c* apparaissent à la fois niées et non niées.

Fonction monoforme et biforme (suite)

Comment vérifier si une fonction est monoforme dans une liste de cube ? => Vérifier que chaque variable n'est présente que dans une polarité (10 ou 01) dans la liste de cube.

Exemple

Soit la fonction f(a,b,c) décrite par cette liste de cube :

$$\begin{bmatrix} 11 & 10 & 10 \end{bmatrix}$$
$$\begin{bmatrix} 01 & 10 & 11 \end{bmatrix}$$
$$\begin{bmatrix} 11 & 11 & 10 \end{bmatrix}$$

f est **monoforme** : *a* est en polarité non niée (01), *b* en polarité niée (10) et *c* en polarité niée (10).

Fonction monoforme et biforme (suite)

Exemple

Soit la fonction f(a,b,c) décrite par cette liste de cube :

$$\begin{bmatrix} 10 & 10 & 10 \end{bmatrix}$$
$$\begin{bmatrix} 01 & 10 & 11 \end{bmatrix}$$
$$\begin{bmatrix} 11 & 11 & 10 \end{bmatrix}$$

f est **biforme** : *a* apparait dans les deux polarités (10 et 01).

Unate Recursive Paradigm (URP)

C'est une méthode d'analyse de fonction booléenne qui permet, entre autre, de :

- vérifier une tautologie,
- calculer le complément d'une fonction,
- déterminer les impliquants premiers.

Principe

- Utiliser les cofacteurs pour décomposer la fonction.
- Profiter des caractéristiques des fonctions et variables monoformes pour l'analyse de la fonction
- Profiter des caractéristiques des fonctions et variables monoformes pour décomposer la fonction.

Unate Recursive Paradigm (URP) : Tautologie

Principe de fonctionnement pour une vérification de tautologie

1. Décomposer la fonction

- On décompose la fonction F en cofacteur : F est une tautologie si ces 2 cofacteurs le sont aussi.
- Quelle variable choisir pour la décomposition ?
  La variable la plus biforme et la plus balancée dans sa biformité ($min(|nombre\ de\ variable\ niee - nombre\ de\ variable\ non\ niee|)$).

Unate Recursive Paradigm (URP) : Tautologie (suite)

Principe de fonctionnement pour une vérification de tautologie

Exemple de décomposition de fonction

Soit la fonction f(a,b,c,d) suivante :

[11 01 10 11] Les variables les plus biformes sont *b* et *c*.
[01 01 10 10] *b* est plus balancée (2 variables niées et 2 non niées)
[10 10 01 11] que *c* (3 variables niées et 1 non niées).
[10 10 10 10] On factorise donc selon *b*.

Selon la méthode présentée en slide 23, nous obtenons :

pour F_b :
[11 11 10 11]
[01 11 10 10]

pour $F_{\bar{b}}$:
[10 11 01 11]
[10 11 10 10]

Unate Recursive Paradigm (URP) : Tautologie

Principe de fonctionnement pour une vérification de tautologie

2. Chercher le cube *All don´t care*

Si un des cubes ne comprend que des *dont´ care* (11), la fonction sera une tautologie.

Exemple

Soit la fonction f(a,b,cd) suivante :

$$\begin{bmatrix} 11 & 01 & 10 & 11 \end{bmatrix}$$
$$\begin{bmatrix} 01 & 01 & 10 & 10 \end{bmatrix}$$
$$\begin{bmatrix} 11 & 11 & 11 & 11 \end{bmatrix}$$
$$\begin{bmatrix} 10 & 10 & 10 & 10 \end{bmatrix}$$

Cette fonction correspond à $b\bar{c} + ab\bar{c}\bar{d} + 1 + \bar{a}\bar{b}\bar{c}\bar{d} = 1$

Unate Recursive Paradigm (URP) : Tautologie

Principe de fonctionnement pour une vérification de tautologie

3. Profiter des caractéristiques des fonctions monoformes

Si un fonction monoforme ne comprend pas le cube *all don't care*, alors cette fonction n'est pas une tautologie

Exemple

Soit la fonction f(a,b,c,d) suivante :

$$
\begin{array}{c}
[\,11 \;\; 01 \;\; 10 \;\; 11\,] \\
[\,10 \;\; 01 \;\; 10 \;\; 10\,] \\
[\,11 \;\; 11 \;\; 10 \;\; 11\,] \\
[\,10 \;\; 11 \;\; 10 \;\; 10\,]
\end{array}
$$

Cette fonction correspond à $b\bar{c} + \bar{a}b\bar{c}\bar{d} + \bar{c} + \bar{a}\bar{c}\bar{d} \neq 1$

Unate Recursive Paradigm (URP) : Tautologie

Principe de fonctionnement pour une vérification de tautologie

4. Chercher des cubes à une variable complémentaires

Si la liste de cube contient deux cubes d'une seule variable compléments l'un de l'autre, alors c'est une tautologie.

Exemple

Soit la fonction f(a,b,c,d) suivante :

$$\begin{bmatrix} 11 & 11 & 11 & 01 \end{bmatrix}$$
$$\begin{bmatrix} 01 & 01 & 10 & 10 \end{bmatrix}$$
$$\begin{bmatrix} 11 & 10 & 01 & 11 \end{bmatrix}$$
$$\begin{bmatrix} 11 & 11 & 11 & 10 \end{bmatrix}$$

Cette fonction vaut $d + ab\bar{c}\bar{d} + \bar{b}c + \bar{d} = ab\bar{c}\bar{d} + \bar{b}c + 1 = 1$.

Unate Recursive Paradigm (URP) : Tautologie

Algorithme pour une vérification de tautologie

```
IsTautology(f in cubelist form){
if (f has an all don´t care cube){
   return 1;
}
else if (f is unate){
   return 0;
}
else if (f has 2 complement cube ){
   return 1;
}
else{
   x = most binate variable in f;
   return IsTautology(fx) && IsTautology(fnot(x));
}}
```

Unate Recursive Paradigm (URP) : Tautologie

Algorithme pour une vérification de tautologie

Exemple

Soit la fonction f(a,b,c,d) suivante :

$$ab + ac + a\bar{b}\bar{c} + \bar{a}$$

Elle est représentée par la liste de cube suivant :

$$\begin{array}{ll}
[01\ \ 01\ \ 11] & \rightharpoonup\ ab \\
[01\ \ 11\ \ 01] & \rightharpoonup\ ac \\
[01\ \ 10\ \ 10] & \rightharpoonup\ a\bar{b}\bar{c} \\
[10\ \ 11\ \ 11] & \rightharpoonup\ \bar{a}
\end{array}$$

La réduction en détails

But : réduire au minimum le nombre de 1 couvert par un impliquant. Cela signifie qu'il faut ajouter une ou des variables dans l'impliquant.

Méthode : une fonction F décrite par une liste de cube qui contient l'impliquant R que l'on souhaite réduire.

Étapes

- Retirer l'impliquant R de la fonction F pour former la fonction $F - \{R\}$.
- Calculer le complément de cette fonction par URP (dédié à la complémentation).
- Faire l'intersection de chaque cube de la fonction $F - \{R\}$ complémentée avec l'impliquant R.
- Faire l'union de tous les cubes qui sont le fruit des intersections.

La réduction en détails (suite)

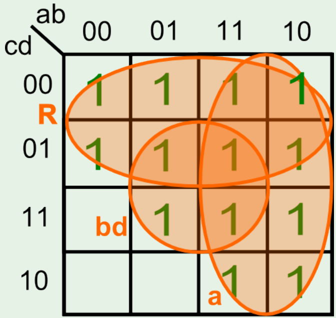
Exemple

Soit la fonction $F(a, b, c, d)$ suivante :

$[\,11\ \ 01\ \ 11\ \ 01\,] \rightharpoonup bd$

$[\,01\ \ 11\ \ 11\ \ 11\,] \rightharpoonup a$

$[\,11\ \ 11\ \ 10\ \ 11\,] \rightharpoonup \bar{c}$ (l'impliquant R à réduire)



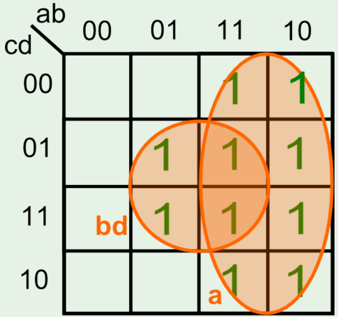
La réduction en détails (suite)

1. Former la fonction $F - \{R\}$

La fonction $F - \{R\}$ est la suivante :

$[\,11\ \ 01\ \ 11\ \ 01\,] \rightharpoonup bd$

$[\,01\ \ 11\ \ 11\ \ 11\,] \rightharpoonup a$



La réduction en détails (suite)

2. Calculer le complément de $F - \{R\}$ par URP

$F - \{R\}$ est la suivante :

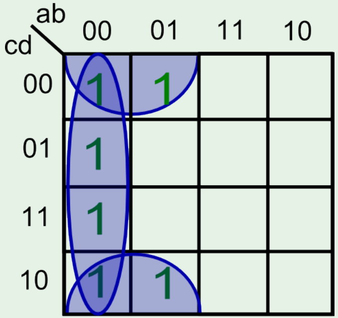
$[\,11\ \ 01\ \ 11\ \ 01\,] \rightharpoonup bd$

$[\,01\ \ 11\ \ 11\ \ 11\,] \rightharpoonup a$

alors son complément peut être décrit par :

$[\,10\ \ 10\ \ 11\ \ 11\,] \rightharpoonup \bar{a}\bar{b}$

$[\,10\ \ 11\ \ 11\ \ 10\,] \rightharpoonup \bar{a}\bar{d}$



La réduction en détails (suite)

Faire l'intersection de chaque cube de la fonction $F - \{R\}$ complémentée avec l'impliquant R.

Pour faire l'intersection de deux cubes décrits en PCN, il suffit de faire un ET logique entre les bits de la description. Ainsi l'intersection de (11 10 11 01) avec (01 11 11 01) est (01 10 11 01). Si jamais vous trouver un '00' dans le fruit de l'intersection, c'est qu'il n'y a pas d'intersection possible.
Cela permet de mettre en évidence les minterms qui sont couvert par R et pas par la fonction $F - \{R\}$.

La réduction en détails (suite)

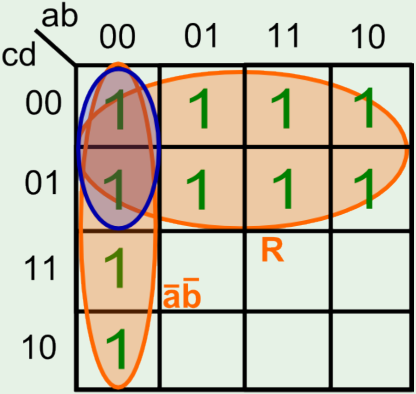
Faire l'intersection de chaque cube de la fonction $F - \{R\}$ complémentée avec l'impliquant R.

Le complément de la fonction $F - \{R\}$ comprend 2 cubes :

$[\,10\ \ 10\ \ 11\ \ 11\,] \rightharpoonup \bar{a}\bar{b}$

$[\,10\ \ 11\ \ 11\ \ 10\,] \rightharpoonup \bar{a}\bar{d}$

L'intersection de l'impliquant $R = [11\,11\,10\,11]$ avec $\bar{a}\bar{b} = [10\,10\,11\,11]$ donne $[\,10\ \ 10\ \ 10\ \ 11\,]$.



La réduction en détails (suite)

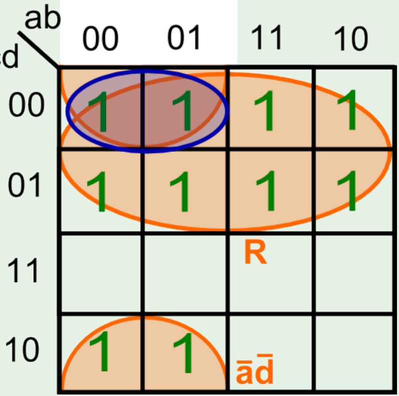
Faire l'intersection de chaque cube de la fonction $F - \{R\}$ complémentée avec l'impliquant R.

Le complément de la fonction $F - \{R\}$ comprend 2 cubes :

$[\,10\ \ 10\ \ 11\ \ 11\,] \rightharpoonup \bar{a}\bar{b}$

$[\,10\ \ 11\ \ 11\ \ 10\,] \rightharpoonup \bar{a}\bar{d}$

L'intersection de l'impliquant $R = [11\,11\,10\,11]$ avec $\bar{a}\bar{d} = [10\,11\,11\,10]$ donne $[\,10\ \ 11\ \ 10\ \ 10\,]$.



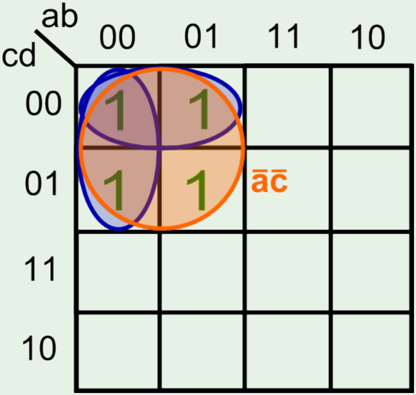
La réduction en détails (suite)

Faire l'union de tous les cubes qui sont le fruit des intersections

Pour faire l'union de deux cubes, il suffit de faire un OU logique entre les bits des cubes. L'union de (11 10 11 01) avec (01 11 11 01) est (11 11 11 01).

L'intersection de R avec la fonction $F - \{R\}$ complémentée donne : [10 10 10 11] et [10 11 10 10].

Pour trouver la version réduite de R, il faut réaliser l'union de ces deux cubes. Nous obtenons [10 11 10 11]



L'expansion en détails

Objectif : agrandir les impliquants d'une fonction de base au maximum pour qu'ils deviennent premiers. On va donc retirer une ou des variables dans l'impliquant.

Étapes

- Établir la fonction complémentaire (OFF-SET) de la fonction à étendre (ON-SET).
- Pour l'impliquant que l'on souhaite étendre, construire la matrice bloquante, qui permet de savoir où on ne peut pas étendre notre impliquant.
- Déterminer un set de variable minimal que doit contenir l'impliquant pour ne pas heurter le OFF-SET.

L'expansion en détails (suite)

Établir la fonction complémentaire (OFF-SET) de la fonction à étendre (ON-SET)

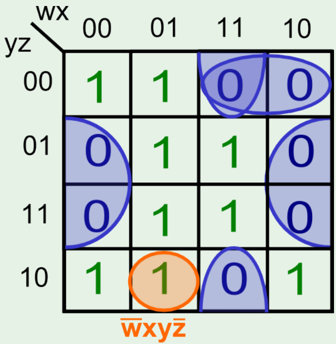
OFF-SET = fonction de base pour laquelle les 0 ont été remplacés par des 1 et inversement, réalisé par l'algorithme URP dédié à la complémentation.

Exemple

Soit la fonction : $f = \bar{w}\bar{y}\bar{z} + xz + \bar{x}y\bar{z} + \bar{w}xy\bar{z}$

Le complément de cette fonction sera : $\bar{f} = \bar{x}z + wx\bar{z} + w\bar{y}\bar{z}$

L'impliquant à étendre est $\bar{w}xy\bar{z}$ (en orange)



L'expansion en détails (suite)

Construire la matrice bloquante

La matrice bloquante (*blocking Matrix*) d'un impliquant R contient :

- Une ligne pour chaque variable (avec sa polarité) de l'impliquant R
- Une colonne pour chaque cube de l'OFF-SET.
- Un '1' dans une case signifie que la variable ligne apparait dans une polarité différente que dans le cube colonne.

Exemple

$f = \bar{w}\bar{y}\bar{z} + xz + \bar{x}y\bar{z} + \bar{w}xy\bar{z}$
$\bar{f} = \bar{x}z + wx\bar{z} + w\bar{y}\bar{z}$
L'impliquant à étendre est $\bar{w}xy\bar{z}$

	$\bar{x}z$	$wx\bar{z}$	$w\bar{y}\bar{z}$
$\bar{w}$	0	1	1
x	1	0	0
y	0	0	1
$\bar{z}$	1	0	0

L'expansion en détails (suite)

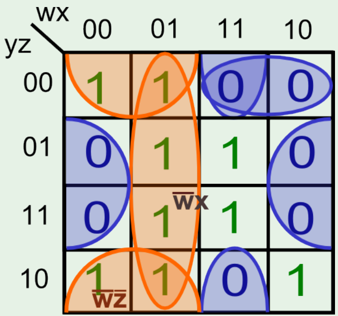
Déduire les variables indispensables de l'impliquant

Objectif : trouver un set minimal de variable pour l'impliquant.

Méthode : choisir un set de variable dans la matrice bloquante de sorte que chaque colonne soit couverte par un 1.

Exemple

	$\bar{x}z$	$wx\bar{z}$	$w\bar{y}\bar{z}$
$\bar{w}$	0	1	1
x	1	0	0
y	0	0	1
$\bar{z}$	1	0	0



Couvertures possibles :

variables $\bar{w}$ et x ($\bar{w}xy\bar{z}$ étendu en $\bar{w}x$)

variables $\bar{w}$ et $\bar{z}$ ($\bar{w}xy\bar{z}$ étendu en $\bar{w}\bar{z}$)

L'irredondance en détails

Objectif : retirer les impliquants ou cubes redondants d'une fonction F.

Principe

Pour chaque cube de la fonction F

- Retirer le cube de la fonction
- Cofactoriser F la fonction par rapport au cube choisi pour obtenir la fonction FS
- Vérifier si la fonction FS vaut 1 (par URP Tautology Checking)
- Si FS vaut 1, le cube est redondant : passer à un autre cube. Si la fonction vaut 0, le cube n'est pas redondant : le remettre dans la fonction.

! Méthode basique. Espresso contient un algorithme plus perfectionné.

L'irredondance en détails (suite)

Retirer le cube S et cofactoriser $F - \{S\}$ par rapport à S

Cofactoriser une fonction par rapport à un cube = remplacer dans la fonction toutes les variables présentes dans le cube par 1 ou 0 en fonction de leur polarité.

Exemple

Soit la fonction $F = \bar{c}d + ad + \bar{a}bc$ et l'impliquant $S = \bar{a}bd$.
On remplace a par 0, b et d par 1 dans la fonction F.
On obtient la fonction $FS = \bar{c} + 0 + c = 1$

L'irredondance en détails (suite)

Vérifier si *FS* est une tautologie

Si *FS* vaut 1, cela signifie que la fonction *FS* vaut 1 pour la même combinaison d'entrées que l'impliquant => l'impliquant est redondant.

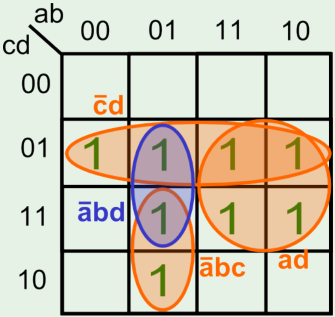
Exemple

$F = \bar{c}d + ad + \bar{a}bc$

$S = \bar{a}bd$

$FS = \bar{c} + 0 + c = 1$

$S = \bar{a}bd$ est déjà couvert par F



Exercices

Enoncé disponible sur Claco. Organisation :

- Travail par binôme
- PC à disposition avec Matlab/Octave durant 5 séances.
- Démonstration par chaque équipe du bon fonctionnement de leur algorithme et des performances obtenues à la 5 ème séance.
- Remise des codes sur Claco. Une question sur son code à l'examen (le code est fourni).

Bibliographie

Giovanni DEMICHELI :
Synthesis and Optimization of Digital Circuits.
McGraw Hill, 1994.

University of Illinois at Urbana-Champaign ROB A. RUTENBAR :
Vlsi cad : Logic to layout.
`https://www.coursera.org/course/vlsicad`, 2013.
En ligne ; consulté le 10/06/2013.

McMullen Sangiovanni-Vincentelli BRAYTON, Hachtel :
Logic Minimization Algorithms for VLSI Synthesis.
Kluwer Academic Press, 1984.

Richard L. RUDELL :
Multiple-Valued Logic Minimization for PLA Synthesis.
Memo No. UCB/ERL M86-65, 1986.