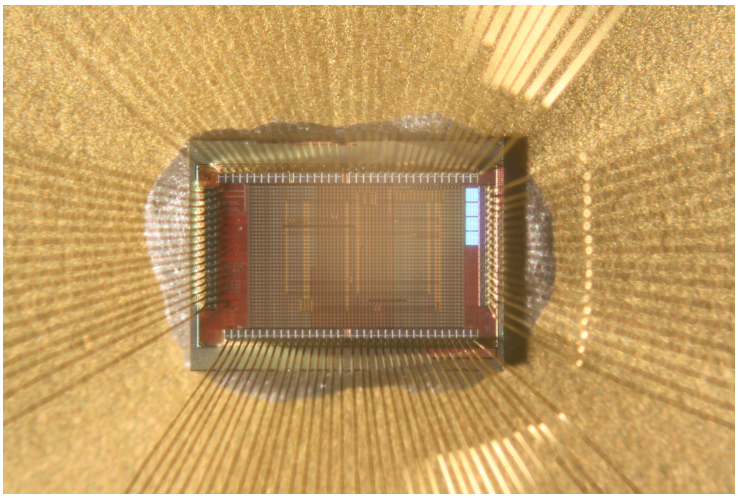
Universitat Politècnica de Catalunya
Dep. Enginyeria Electrònica - ETSEIB

Quadern de Teoria –

Sistemes Combinacionals



Introducció a les

Funcions Lògiques

i a la

Minimització d'Expressions

Álvaro Gómez i Salvador Manich
Grup de Recerca de Qualitat en els Sistemes Electrònics - QinE

Índex

Índex de figures

Índex de taules

Capítol 1

Introducció

En aquest quadern es tractarà l'àlgebra que s'empra en el disseny de sistemes digitals atemporals. Un sistema atemporal és aquell en què la seva resposta no depèn del temps i que en conseqüència reacciona de manera instantània i única als impulsos rebuts a l'entrada.

L'àlgebra dels sistemes digitals és l'anomenada àlgebra de commutació, que és una sub-àlgebra dins de l'anomenada àlgebra de Boole. En aquesta es treballa amb variables binàries que únicament prenen dos estats (o valors), "0" (fals) o "1"(cert). Aquesta és la mínima informació que es pot emmagatzemar en una variable[1] [1].

Quan volem representar magnituds numèriques amb més de dos estats hem d'emprar més d'una variable binària, e. g. per indicar la temperatura atmosfèrica.

Exemple 1 Si volem representar la temperatura des del 0 °C fins als 15 °C, amb una codificació binària ho podem fer amb quatre variables binàries tal com es mostra a la Taula 1.1. En aquesta, la codificació binària emprada és la natural.

Un conjunt de variables binàries que codifiquen la mateixa informació s'agrupen en vectors (binaris)[2]. Les components del vectors a la seva vegada són variables binàries, i.e. $X = \{x_{n-1}, ..., x_1, x_0\}$ que és un vector de n components on la component de l'esquerra, x_{n-1} conté el bit més significatiu i la de la dreta x_0 el bit menys significatiu.

La representació numèrica de les magnituds es pot fer en altres codificacions diferents de la binària, les més conegudes són l'octal, la decimal i l'hexadecimal[3]. En la Taula

[1] En aquest document les variables binaries es representen amb una lletra itàlica, e.g. x_b, on el subíndex indica el tipus de codificació o sense si el context ja ho deixa clar, i. e. x.

[2] Es representen habitualment per una lletra majúscula, e. g. X_b on el subíndex indica el tipus de codificació o sense si el context ho deixa clar. En alguns casos, si no dona lloc a confusió, els vectors també es poden representar per lletres minúscules com x.

[3] Les variables i els vectors que contenen magnituds emprant aquestes codificacions es representen

Taula 1.1: Codificació binària natural de setze valors de temperatura.

Temperatura [°C]	Codificació binària natural x_3	x_2	x_1	x_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

1.2 es mostren els diferents valors que poden prendre les variables en els quatre tipus de codificació esmentats.

Taula 1.2: Conjunt de valors que prenen les variables representades en codificació hexadecimal, decimal, octal i binari.

x_h	x_d	x_o	x_b
0	0	0	0
1	1	1	1
2	2	2	
3	3	3	
4	4	4	
5	5	5	
6	6	6	
7	7	7	
8	8		
9	9		
A			
B			
C			
D			
E			
F			

Totes les variables codificades en una notació de més pes (e.g. hexadecimal) tenen una representació equivalent en un vector codificat en una notació de menys pes (e.g. binària), i a l'inrevés. Existeixen funcions de conversió d'una notació a qualsevol altre i en aquest document s'assumeix que el lector les coneix. Només s'explicitaran aquelles que pel context ho facin necessari.

Exemple 2 En la Taula 1.3 es mostren algunes conversions dels tipus següent: A) variable decimal a vector binari, B) vector binari a variable hexadecimal, C) vector binari a vector decimal i D) vector octal a vector binari.

Quan es codifiquen magnituds emprant vectors, el més habitual és que el rang de representació sigui finit. Això vol dir que el nombre de components del vector està prefixat a un de concret. Aquesta és, per exemple, la manera de treballar dels microprocessadors en la seva arquitectura més interna. Això comporta que algunes conversions puguin donar lloc a desbordaments (incorreccions), com la que es mostra en el següent exemple.

per les lletres minúscula o majúscula respectivament amb un subíndex que especifica el tipus de codificació, si el context no ho deixa clar. Per exemple, la variable x_h conté informació codificada en notació hexadecimal i el vector X_h conté variables hexadecimals en les seves components.

Taula 1.3: Diferents tipus de conversions entre variables i vectors.

	A	
x_d	$\rightarrow$	$Y_b = \{y_3, y_2, y_1, y_0\}$
3		0011
9		1001
7		0111
	B	
$X_b = \{x_3, x_2, x_1, x_0\}$	$\rightarrow$	y_h
1010		A
0010		2
1111		F
	C	
$X_b = \{x_4, x_3, x_2, x_1, x_0\}$	$\rightarrow$	$Y_d = \{y_1, y_0\}$
10011		19
00101		05
11101		29
	D	
$X_o = \{x_1, x_0\}$	$\rightarrow$	$Y_b = \{y_4, y_3, y_2, y_1, y_0\}$
23		10011
05		00101
35		11101

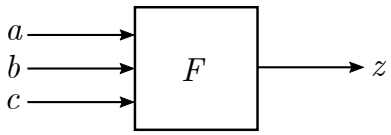


Figura 1.1: Circuit combinacional que implementa per una funció lògica de tres variables d'entrada.

Exemple 3

Suposem que volem convertir el següent número $(234)_d$ (codificació decimal) a una codificació binària[4]. La conversió ens donarà el número 11101010 que requeriria un vector de vuit components per emmagatzemar-lo. Ara bé, imaginem que un sistema digital concret únicament disposa d'un vector de quatre components, i que aquestes emmagatzemen la part menys significativa del valor. Si posteriorment intentem recuperar el valor numèric en decimal obtindríem el valor numèric $(10)_d$, tal com es pot veure en la Taula 1.4.

Taula 1.4: Desbordament per manca de variables binàries en una conversió.

$X_d \rightarrow Y_{4b} \rightarrow X_d$

$X_d =$	Codificacio decimal $(234)_d$
	Conversio a binari 11101010
$Y_{4b} =$	Truncament a 4 LSB ␣␣␣␣1010
$X_d =$	Conversio a decimal $(10)_d$

Els sistemes combinacionals[5] estan descrits per funcions lògiques (combinacionals). Una funció lògica és aquella que genera una variable binaria de sortida a partir d'un nombre de variables d'entrada concret. Per exemple, la funció $z = F(a, b, c)$ anuncia la funció F de tres variables d'entrada, que descriurà el circuit combinacional[6] indicat a la Figura 1.1.

Llibres de Referència

Hi ha diversa literatura clàssica que tracta els temes presentats en aquest quadern d'una manera extensa. Pels que desitgeu aprofundir en el seu contingut us remetem

[4] Habitualment la codificació és binària natural, a no ser que es digui el contrari.
[5] Un sistema combinacional està format per un conjunt de funcions lògiques (combinacionals).
[6] Un circuit combinacional és la realització (esquemàtica o física) d'una funció lògica o sistema combinacional.

als llibres indicats a continuació.

- "Introduction to Digital Logic Design" [2]. Aquest és el llibre de referència emprat en l'elaboració d'aquest quadern.
- "Análisis y Diseño de Circuitos Lógicos Digitales" [3].
- "Digital Design: Principles and Practices" [4].

Capítol 2

Especificació de Funcions Lògiques

Les dues maneres més simples d'especificar les funcions lògiques són per mitjà de Taules de la Veritat (TV) o d'Expressions Lògiques (EL). En aquest apartat parlarem primer de les TV i deixarem per més endavant les EL.

2.1 Taula de la Veritat

Una TV és una enumeració (tabular) de tots els valors que pot prendre una funció lògica per qualsevol combinació dels seus valors d'entrada. E.g. suposem que el vector X_{4b} conté els nombres d'entre 0 al 15. En la Taula 2.1 podem veure la TV de la funció $P(X_{4b})$ que detecta els nombres primers.

Si n és el nombre de variables de la funció, una TV té un nombre màxim de files igual a 2^n, que correspon a totes les combinacions binàries de les variables. En conseqüència, quan el nombre de variables d'entrada de la funció augmenta, la definició d'aquesta a partir d'una TV es fa inviable, a no ser que s'empri algun tipus de software CAE (Computer Aided Engineering) de suport. Per exemple, si la mateixa funció $P()$ la volguéssim estendre a 32 variables, la dimensió de la TV seria de màxim $2^{32} = 4.294.967.296$ files.

Exemple 4 Si volem descriure totes les possibles funcions d'una i dues variables, només cal que generem totes les TVs diferents possibles amb aquest nombre de variables. Aquestes es poden veure en la Taula 2.2

De funcions d'una variable n'hi ha quatre en total. Una d'aquestes és rellevant, la NOT, també anomenada complement o inversió, que és una primitiva de l'àlgebra de commutació. Les altres tres funcions són la identitat, $I()$ i les dues funcions constants a "0" i a "1" que són degenerades* ja que no depenen de la variable a.

Taula 2.1: Taula de la veritat de la funció que detecta els nombres primers. En la versió compacta, les variables que no provoquen un canvi en l'estat de la funció es poden especificar amb una X (no-importa), indicant que la funció de sortida pren el mateix valor per qualsevol valor de la variable.

x_3	x_2	x_1	x_0	$P(X_{4b})$
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

Versió més compacta

x_3	x_2	x_1	x_0	$P(X_{4b})$
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	X	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	X	0

Taula 2.2: Totes les funcions lògiques possibles d'una i dues variables.

a	K_0^*	$I(a)$	NOT(a)	K_1^*
0	0	0	1	1
1	0	1	0	1

a	b	K_0^*	NOR(a,b)		NOT$^*(a)$
0	0	0	1	0	1
0	1	0	0	1	1
1	0	0	0	0	0
1	1	0	0	0	0

Continuació

a	b		NOT$^*(b)$	XOR(a,b)	NAND(a,b)
0	0	0	1	0	1
0	1	0	0	1	1
1	0	1	1	1	1
1	1	0	0	0	0

Continuació

a	b	AND(a,b)	XNOR(a,b)	$I^*(b)$	
0	0	0	1	0	1
0	1	0	0	1	1
1	0	0	0	0	0
1	1	1	1	1	1

Continuació

a	b	$I^*(a)$		OR(a,b)	K_1^*
0	0	0	1	0	1
0	1	0	0	1	1
1	0	1	1	1	1
1	1	1	1	1	1

De funcions de dues variables n'hi ha setze en total. Dues d'aquestes són molt rellevants, AND i OR, ja que són primitives de l'àlgebra de commutació. Hi ha altres funcions molt emprades que tenen noms coneguts: NOR, XOR, NAND i XNOR però que no són primitives donat que es poden obtenir combinant algunes de les primitives anteriors. Per exemple, $\text{NAND}(a,b) \equiv \text{NOT}(\text{AND}(a,b))$ o $\text{XOR}(a,b) \equiv \text{OR}(\text{AND}(\text{NOT}(a),b),\text{AND}(a,\text{NOT}(b)))$. La resta de funcions, o són degenerades (funció de menys de dues variables) o no tenen cap interès especial.

Com s'ha pogut observar en la Taula 2.2, per a la generació de totes les funcions possibles d'un nombre de variables d'entrada, s'ha aplicat una doble combinatòria. Inicialment s'han enumerat tots els valors possibles de les variables d'entrada, 2 per una variable i 4 per dues variables, que constitueixen les files de les TVs. Per altre cantó, s'han enumerat totes les combinacions possibles dels valors de les funcions començant pel tot a 00...0 i acabant pel tot a 11...1 (columnes de sortida de les TVs). Extrapolant, doncs, aquesta regla generadora, es pot afirmar que el nombre total de funcions de n variables és igual a $N = 2^{2^n}$ (una doble potència de 2), que per $n = 1, 2, 3, 4...$ dóna $N = 4, 16, 256, 65.536, ...$.

L'especificació d'una funció per mitjà d'una TV té la propietat de ser única. No hi ha dues TVs diferents que representin la mateixa funció. Més endavant veurem una altra manera d'especificar funcions que sí presenta multiplicitat de representacions, però en el cas de les TVs això no passa.

La TV pot estar parcialment especificada, si resulta que hi ha combinacions de les variables d'entrada que no es donen mai. En aquest cas, en les combinacions d'entrada inexistents es pot indicar la sortida de la funció com a "no-importa", emprant el símbol X.

Exemple 5 Es vol especificar una funció lògica que activi un avís sonor en un vehicle de dues places, si els ocupants no tenen lligat el cinturó de seguretat. Es disposa de quatre variables: s_0 i s_1 ens indiquen si els seients tenen ocupant assegut quan prenen el valor "1", i c_0 i c_1 si el cinturó de seguretat està enclavat quan prenen aquest mateix valor. La definició de la funció que activa el senyal d'avís, $a = A(s_0, s_1, c_0, c_1)$, es troba a Taula 2.3.

Si suposem que un seient sense ocupant no tindrà mai el cinturó de seguretat enclavat llavors, en totes aquelles combinacions on el seient té 0 i el cinturó 1 podrem assignar X a la funció, donat que aquesta mai rebrà la combinació d'entrades concreta. Alternativament, podem pensar que encara que estigui el cinturó enclavat, si no hi ha ocupant, no hi haurà cap risc per la seguretat i per tant "no-importa" si l'avís s'activa o no.

A la part dreta de la Taula 2.3 s'inclou una alternativa d'especificació de TV que permet reduir la mida d'aquesta. Enlloc de marcar les posicions de la TV amb X simplement s'eliminen les files corresponents.

Aclariment respecte de X. Quan una funció està parcialment especificada signifi-

Taula 2.3: TV parcialment especificada.

$A(s_0, s_1, c_0, c_1)$

s_0	s_1	c_0	c_1	a
0	0	0	0	0
0	0	0	1	X
0	0	1	0	X
0	0	1	1	X
0	1	0	0	1
0	1	0	1	0
0	1	1	0	X
0	1	1	1	X
1	0	0	0	1
1	0	0	1	X
1	0	1	0	0
1	0	1	1	X
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Alternativa d'especificació

s_0	s_1	c_0	c_1	a
0	0	0	0	0
0	1	0	0	1
0	1	0	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

ca que durant el procés de síntesi de la funció en forma de circuit combinacional, aquest fixarà un valor de 0 o 1 per a aquella combinació d'entrades segons hagi estat realitzada la síntesi, sense importar si és un o l'altre valor. Com es veurà més endavant, en la síntesi de funcions, això té unes conseqüències molt interessants durant la simplificació dels circuits combinacionals.

Si bé la TV és una manera acurada i simple d'especificar funcions lògiques, té l'inconvenient de la baixa compacitat. La seva mida creix exponencialment amb el nombre de variables d'entrada, tal com ja s'ha indicat anteriorment. Per aquest motiu l'us de les TVs es restringeix a les funcions de poques variables, en paper fins a 5 i en ordinador típicament fins a 10 variables.

Una alternativa més flexible d'especificació de funcions lògiques és emprar expressions lògiques (combinacionals)(EL). Amb aquestes la notació és més compacte i alhora es poden especificar funcions a partir d'altres funcions.

2.2 Expressió Lògica

En l'àlgebra de commutació es defineixen unes funcions primitives $\{\text{AND}, \text{OR}, \text{NOT}\}$ que junt amb els elements $\{0, 1\}$ componen aquesta i que verifiquen els axiomes, postulats i teoremes derivats de l'àlgebra de Boole [2](“A definition of Boolean Al-

gebra", sec. 3.6, pg. 160)[1]. La definició de les funcions primitives la podeu trobar a la Taula 2.2. En la Taula 2.4 es poden veure alguns dels símbols[2] més emprats (expressions) per representar aquestes funcions primitives[3].

Taula 2.4: Símbols més emprats per a les funcions primitives.

AND(a, b)

a AND b	$a \cdot b$	$a \wedge b$

OR(a, b)

a OR b	$a + b$	$a \vee b$

NOT(a)

NOT a	$\overline{a}$	$\neg a$
	a'	

A partir de les funcions primitives es poden elaborar expressions lògiques (EL) que especifiquin funcions més complexes, a base de combinar aquestes. Algunes d'elles, prenen noms pel fet de ser molt emprades, com les indicades a la Taula 2.2.

Així doncs, les EL són una segona manera d'especificar funcions lògiques. Ara be, una mateixa funció lògica pot tenir moltes ELs equivalents. Les regles de construcció són les mateixes que en l'àlgebra ordinària[4] i es poden emprar parèntesis per tal d'especificar l'ordre de realització dels càlculs (precedència de les operacions) [2]("Expressions, Functions, and Circuits / Well-Formed Expressions", pg. 179) [8]. A fi de simplificar l'us dels parèntesis s'estableix una precedència en el càlcul de les funcions primitives que és de més a menys: NOT, AND, OR.

Exemple 6 Troba ELs equivalents a l'expressió $z = \overline{c} \cdot (a \cdot b + \overline{a} \cdot c)$ i determina el nombre de nivells. Troba també la TV de la funció.

1. L'expressió inicial és $\overline{c} \cdot (a \cdot b + \overline{a} \cdot c)$. El nombre de nivells[5] requerit per fer el càlcul és $\{\overset{1}{\Rightarrow} \overline{c} \cdot (u + v) \overset{2}{\Rightarrow} \overline{c} \cdot w \overset{3}{\Rightarrow} z\}$ de 3.
2. Apliquem la propietat distributiva de la AND respecte de la OR $\rightarrow a \cdot b \cdot \overline{c} + \overline{a} \cdot c \cdot \overline{c}$. El nombre de nivells és $\{\overset{1}{\Rightarrow} u + v \overset{2}{\Rightarrow} z\}$ de 2.
3. A l'expressió 2 apliquem la propietat del complement en la AND[6] i la pro-

[1] Es poden consultar també les referències precedents següents, [5] [6] [7].
[2] Cal no confondre els símbols $+$ i $\cdot$ amb la suma i el producte de l'aritmètica.
[3] En el llenguatge de programació C s'empren els símbols $(\&\&, \|, !)$ per a l'avaluació de condicions lògiques i $(\&, |, \sim)$ per al càlcul de lògica bit a bit.
[4] Cal tenir present que un subconjunt dels axiomes de l'àlgebra de Boole s'assemblen als de l'algebra ordinària si s'assimilen la AND com el producte i la OR com la suma. Ara be, hi ha un altre subconjunt dels axiomes que és diferent. Més endavant es resaltaran les diferències segons les necessitats de l'explicació.
[5] En el càlcul dels nivells d'una expressió no es compten les funcions NOT.
[6] $a \cdot \overline{a} = 0$

pietat d'absorció en la AND[7], ens dóna $\rightarrow a \cdot b \cdot \overline{c}$. El nombre de nivells és $\{\overset{1}{\Rightarrow} z\}$ de 1.

4. Prenem 1 i emprant la propietat de la involució[8] fem una doble negació $\rightarrow \overline{\overline{c} \cdot (a \cdot b + \overline{a} \cdot c)}$. El nombre de nivells és el mateix, 3.

5. Ara apliquem la llei de Morgan[9] i la d'involució allà on calgui $\rightarrow \overline{c + \overline{(a \cdot b + \overline{a} \cdot c)}}$. El nombre de nivells segueix sent 3.

6. Repetint la mateixa transformació obtenim $\rightarrow \overline{c + \overline{\overline{(a \cdot b)} \cdot \overline{(\overline{a} \cdot c)}}}$, de 3 nivells.

7. $\rightarrow \overline{c + (\overline{a} + \overline{b}) \cdot (a + \overline{c})}$, de 3 nivells.

8. Apliquem ara la propietat distributiva de la OR respecte de la AND[10] $\rightarrow \overline{(\overline{a} + \overline{b} + c) \cdot (a + \overline{c} + c)}$ que és de 2 nivells. $\{\overset{1}{\Rightarrow} \overline{u \cdot v} \overset{2}{\Rightarrow} z\}$.

9. Apliquem la propietat del complement en la OR[11], de l'absorció en la OR[12] i de la identitat en la AND[13], $\rightarrow \overline{(\overline{a} + \overline{b} + c)}$. És de 1 nivell.

10. Apliquem la llei de Morgan i la d'involució, $\rightarrow a \cdot b \cdot \overline{c}$ que és la mateixa expressió que 3.

    Com que totes les ELs anteriors són equivalents, prenent qualsevol d'elles podem construir la TV a base de fixar les variables a tots els valors possibles i obtenir el resultat de l'expressió. La podeu veure a la Taula 2.5.

Taula 2.5: Taulta de la veritat de la funció $z = \overline{c} \cdot (a \cdot b + \overline{a} \cdot c)$.

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

En forma compacta

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

[7] $a \cdot 0 = 0$
[8] $\overline{\overline{a}} = a$
[9] $\overline{a \cdot b} = \overline{a} + \overline{b}$ [2]("Fundamental Properties / Informal Proofs", pg. 173).
[10] Aquesta propietat no es compleix en l'àlgebra ordinaria: $a + (b \cdot c) = (a + b) \cdot (a + c)$.
[11] $a + \overline{a} = 1$
[12] $a + 1 = 1$
[13] $a \cdot 1 = a$

2.3 Mintermes i Maxtermes

El minterme[14] és una funció especial que genera únicament un 1 per una combinació de les seves variables d'entrada (presenta el nombre mínim de 1ns possible).

Exemple 7 Escriu les TVs i les ELs de tots els mintermes possibles de les tres variables binàries $\{a, b, c\}$.

Taula 2.6: Tots els mintermes possibles de tres variables.

	a	b	c	m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7
0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
2	0	1	0	0	0	1	0	0	0	0	0
3	0	1	1	0	0	0	1	0	0	0	0
4	1	0	0	0	0	0	0	1	0	0	0
5	1	0	1	0	0	0	0	0	1	0	0
6	1	1	0	0	0	0	0	0	0	1	0
7	1	1	1	0	0	0	0	0	0	0	1

En la Taula 2.6 teniu les TVs. Les ELs són les següents:

$$m_0 = \overline{a} \cdot \overline{b} \cdot \overline{c} \qquad m_4 = a \cdot \overline{b} \cdot \overline{c}$$
$$m_1 = \overline{a} \cdot \overline{b} \cdot c \qquad m_5 = a \cdot \overline{b} \cdot c$$
$$m_2 = \overline{a} \cdot b \cdot \overline{c} \qquad m_6 = a \cdot b \cdot \overline{c}$$
$$m_3 = \overline{a} \cdot b \cdot c \qquad m_7 = a \cdot b \cdot c$$

L'expressió d'un minterme està formada per un terme producte (AND) i conté tots els literals[15] possibles. Es representa amb una ema minúscula, el subíndex de la qual indica quins dels literals estan negats. Per una funció de n variables,

$$m_i = \dot{x}_{n-1} \cdot \cdots \cdot \dot{x}_j \cdot \cdots \cdot \dot{x}_1 \cdot \dot{x}_0 \tag{2.1}$$

on el literal $\dot{x}_j$ estarà negat si en la representació binària del subíndex i apareix un 0 en la posició j.

El maxterme[16] és una funció especial complementària al minterme. Presenta el màxim nombre de 1ns per una funció, el que significa que té únicament un 0 en la seva TV.

[14] [2](“Basic Structures / Minterms”, pg. 202).

[15] Un literal és la representació d'una variable en el seu estat negat o no-negat present en una expressió lògica. Una variable que apareix diferents cops dins d'una mateixa expressió dona lloc a diversos literals.

[16] [2](“Basic Structures / Maxterms”, pg. 206).

Exemple 8 Escriu les TVs i les ELs de tots els maxtermes possibles de les tres variables binàries $\{a, b, c\}$.

Taula 2.7: Tots els max-termes possibles de tres variables.

	a	b	c	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7
0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	0	1	1	1	1	1	1
2	0	1	0	1	1	0	1	1	1	1	1
3	0	1	1	1	1	1	0	1	1	1	1
4	1	0	0	1	1	1	1	0	1	1	1
5	1	0	1	1	1	1	1	1	0	1	1
6	1	1	0	1	1	1	1	1	1	0	1
7	1	1	1	1	1	1	1	1	1	1	0

En la Taula 2.7 teniu les TVs. Les ELs són les següents:

$$M_0 = a + b + c \qquad M_4 = \overline{a} + b + c$$
$$M_1 = a + b + \overline{c} \qquad M_5 = \overline{a} + b + \overline{c}$$
$$M_2 = a + \overline{b} + c \qquad M_6 = \overline{a} + \overline{b} + c$$
$$M_3 = a + \overline{b} + \overline{c} \qquad M_7 = \overline{a} + \overline{b} + \overline{c}$$

L'expressió d'un maxterme està formada per un terme suma[17] (OR) i conté tots els literals possibles. Es representa amb una ema majúscula, el subíndex de la qual indica quins dels literals estan negats. Per una funció de n variables,

$$M_i = \dot{x}_{n-1} \cdots + \dot{x}_j \cdots + \dot{x}_1 + \dot{x}_0 \tag{2.2}$$

on el literal $\dot{x}_j$ estarà no-negat si en la representació binària del subíndex i apareix un 0 en la posició j.

2.3.1 Expressions Canòniques

En l'especificació d'una funció lògica a partir d'una EL, existeixen les anomenades expressions canòniques (EC) que són expressions úniques, de dos nivells i que estan formades a partir de la unió de mintermes o de la intersecció de maxtermes. Quan es construeix amb la unió de mintermes s'obté la EC suma-de-productes (SOP) o també anomenada forma I (EC-I). Quan es forma a partir de la intersecció de maxtermes s'obté la EC producte-de-sumes (POS) o també anomenada forma II (EC-II). Com que són formes úniques, cada funció lògica només tindrà aquestes dues úniques expressions canòniques associades, la EC-I i la EC-II[18].

[17] El nom "suma" es pren per la forma del símbol + però no té cap relació amb la suma aritmètica. Per no caure en confusió cal tenir en conte el context en el que es fa servir aquest nom.

[18] [2]("Minimal Forms / Canonical Forms", pg. 279).

Exemple 9 Determina les EC-I i EC-II de la funció especificada per la TV de la Taula 2.1.

Per determinar la EC-I cal, en primer lloc, identificar els mintermes de la funció. En la Taula 2.8 els podeu veure.

Taula 2.8: Identificació dels min-termes continguts en la funció de detecció dels nombres primers.

x_3	x_2	x_1	x_0	$P(X_{4b})$	m_1	m_3	m_5	m_7	m_{11}	m_{13}
0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0
0	0	1	1	1	0	1	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0
0	1	0	1	1	0	0	1	0	0	0
0	1	1	0	0	0	0	0	0	0	0
0	1	1	1	1	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0
1	0	1	1	1	0	0	0	0	1	0
1	1	0	0	0	0	0	0	0	0	0
1	1	0	1	1	0	0	0	0	0	1
1	1	1	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

Si fem la OR de tots els mintermes obtindrem l'expressió lògica de la funció de detecció dels nombres primers. Aquesta és la forma EC-I,

$$\begin{aligned} P(X_{4b}) &= m_1 + m_3 + m_5 + m_7 + m_{11} + m_{13} \\ &= \overline{x_3} \cdot \overline{x_2} \cdot \overline{x_1} \cdot x_0 + \overline{x_3} \cdot \overline{x_2} \cdot x_1 \cdot x_0 + \overline{x_3} \cdot x_2 \cdot \overline{x_1} \cdot x_0 \\ &+ \overline{x_3} \cdot x_2 \cdot x_1 \cdot x_0 + x_3 \cdot \overline{x_2} \cdot x_1 \cdot x_0 + x_3 \cdot x_2 \cdot \overline{x_1} \cdot x_0 \end{aligned}$$

Per a trobar la forma EC-II hem d'identificar les maxtermes continguts en la funció. En la Taula 2.9 els podeu trobar. Agrupant-los tots amb una AND obtindrem l'expressió EC-II de la funció,

$$\begin{aligned} P(X_{4b}) &= M_0 \cdot M_2 \cdot M_4 \cdot M_6 \cdot M_8 \cdot M_9 \cdot M_{10} \cdot M_{12} \cdot M_{14} \cdot M_{15} \\ &= (x_3 + x_2 + x_1 + x_0) \cdot (x_3 + x_2 + \overline{x_1} + x_0) \cdot (x_3 + \overline{x_2} + x_1 + x_0) \\ &\cdot (x_3 + \overline{x_2} + \overline{x_1} + x_0) \cdot (\overline{x_3} + x_2 + x_1 + x_0) \cdot (\overline{x_3} + x_2 + x_1 + \overline{x_0}) \\ &\cdot (\overline{x_3} + x_2 + \overline{x_1} + x_0) \cdot (\overline{x_3} + \overline{x_2} + x_1 + x_0) \cdot (\overline{x_3} + \overline{x_2} + \overline{x_1} + x_0) \\ &\cdot (\overline{x_3} + \overline{x_2} + \overline{x_1} + \overline{x_0}) \end{aligned}$$

Taula 2.9: Identificació de tots els max-termes continguts en la funció de detecció dels nombres primers.

$P(X_{4b})$	M_0	M_2	M_4	M_6	M_8	M_9	M_{10}	M_{12}	M_{14}	M_{15}
0	0	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
0	1	0	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
0	1	1	0	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
0	1	1	1	0	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1
0	1	1	1	1	0	1	1	1	1	1
0	1	1	1	1	1	0	1	1	1	1
0	1	1	1	1	1	1	0	1	1	1
1	1	1	1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1	0	1	1
1	1	1	1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	0

De l'exemple anterior es pot veure que la forma general de EC-I per a una funció de n variables és,

$$\begin{aligned} \text{EC-I} &\triangleq \sum_i m_i \quad | \quad m_i \in \mathcal{F}_1 \\ &\triangleq \sum_i \dot{x}_{n-1,i} \cdot \cdots \cdot \dot{x}_{0,i} \end{aligned} \tag{2.3}$$

on el sumatori es pren com el símbol de seqüència de la OR lògica, per paral·lelisme amb el símbol "+". La OR de seqüència s'estén a tots els mintermes que pertanyen al conjunt de 1ns de la funció.

Semblantment, la forma general de la EC-II és,

$$\begin{aligned} \text{EC-II} &\triangleq \prod_i m_i \quad | \quad m_i \in \mathcal{F}_0 \\ &\triangleq \prod_i \dot{x}_{n-1,i} + \cdots + \dot{x}_{0,i} \end{aligned} \tag{2.4}$$

on el producte de seqüència es pren com el símbol de seqüència de la AND lògica, per paral·lelisme amb el símbol "·". La AND de seqüència s'estén a tots els maxtermes que pertanyen al conjunt de 0s de la funció.

Les expressions canòniques tenen també una representació compacta emprant direc-

tament la codificació decimal dels índexs i. Aquestes són,

$$\text{EC-I} \Rightarrow \sum(0,1,\cdots,i) \quad | \quad F(i)=1$$
$$\text{EC-II} \Rightarrow \prod(0,1,\cdots,i) \quad | \quad F(i)=0$$

Exemple 10 Escriure, de l'Exemple 9, les formes compactes de les ECs.

Per a la EC-I aquesta és,

$$P(X_{4b}) = \sum(1,3,5,7,11,13)$$

I per a la EC-II és,

$$P(X_{4b}) = \prod(0,2,4,6,8,9,10,12,14,15)$$

Una expressió lògica es pot considerar com un primer pas en la implementació d'un circuit lògic. A partir de l'expressió es tenen indicacions de com connectar les portes lògiques entre si, si és que es fa una implementació d'aquesta mena[19].

És per aquest motiu que les funcions lògiques parcialment especificades (amb "no-importa") no poden ser descrites adequadament amb ELs. Cal tenir en compte que si intentem especificar una funció lògica, que conté no-importes, amb una EL immediatament estarem fent una assignació arbitraria dels no-importes a algun valor "1" o "0". L'única forma d'expressió que permet especificar "no-importes"és la forma compacta.

En el següent exemple es poden veure les consideracions que s'han fet en relació als no-importes en les expressions lògiques, en particular en les canòniques.

Exemple 11 Un sistema conté dues funcions lògiques, z, w que detecten si en una entrada d'un peatge hi ha un vehicle i de quin tipus. Reben tres entrades ve, cx, ca. Si les tres estan a 0 és que no hi ha vehicle, $z=0, w=0$. Si $ve=1$ és que hi ha un vehicle lleuger, que serà una moto si les altres dues variables estan a 0, $z=1, w=0$. Si $ve=cx=1$ és que és un cotxe, per tant també un vehicle lleuger $z=1, w=0$, mentre que si les tres variables $ve=cx=ca=1$ és que és un camió, per tant un vehicle pesat i $z=1, w=1$. Tal com està muntat el sistema, les variables no rebran mai cap de les altres combinacions.

Especifiquem les funcions amb TVs, amb les expressions compactes i amb les EC-I i EC-II. Determinem les assignacions fetes per les ECs en els no-importes.

En la Taula 2.10 hi han la TVs de les funcions. Les expressions compactes de la forma EC-I són les següents:

$$z = \sum(4,6,7) + \Delta(1,2,3,5)$$
$$w = \sum(7) + \Delta(1,2,3,5)$$

[19] [2]("Basic Structures / Expression-Circuit Correspondence", pg. 204).

Taula 2.10: Taula de la veritat de les funcions detectores de vehicles.

ve	cx	ca	z	w
0	0	0	0	0
0	0	1	X	X
0	1	0	X	X
0	1	1	X	X
1	0	0	1	0
1	0	1	X	X
1	1	0	1	0
1	1	1	1	1

on el terme $\Delta()$ indica els mintermes que es veuen afectats pels "no-importes". Les expressions compactes de la forma EC-II són les següents:

$$z = \prod(0) \cdot \Delta(1,2,3,5)$$
$$w = \prod(0,4,6) \cdot \Delta(1,2,3,5)$$

on el terme $\Delta()$ indica els maxtermes que es veuen afectats pels "no-importes". Les expressions canòniques regulars EC-I són,

$$z = ve \cdot \overline{cx} \cdot \overline{ca} + ve \cdot cx \cdot \overline{ca} + ve \cdot cx \cdot ca$$
$$w = ve \cdot cx \cdot ca$$

que si les avaluem per tots els valors possibles de les seves variables d'entrada obtindrem la TV de la Taula 2.11, on els no-importa han quedat assignats a "0" (veure els valors en cursiva).

Taula 2.11: Taula de la veritat obtinguda directament de les expressions canòniques.

			EC-I		EC-II	
ve	cx	ca	z	w	z	w
0	0	0	0	0	0	0
0	0	1	*0*	*0*	*1*	*1*
0	1	0	*0*	*0*	*1*	*1*
0	1	1	*0*	*0*	*1*	*1*
1	0	0	1	0	1	0
1	0	1	*0*	*0*	*1*	*1*
1	1	0	1	0	1	0
1	1	1	1	1	1	1

Semblantment, les expressions canòniques regulars EC-II són,

$$z = ve + cx + ca$$
$$w = (ve + cx + ca) \cdot (\overline{ve} + cx + ca) \cdot (\overline{ve} + \overline{cx} + ca)$$

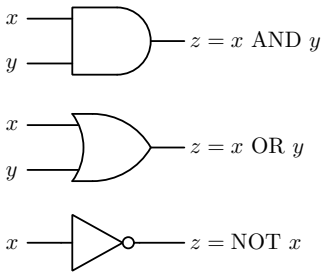


Figura 2.1: Símbols normalitzats de les portes lògiques bàsiques.

Avaluant l'expressió amb tots els valors possibles de les variables d'entrada i obtindrem la TV que es mostra a la Taula 2.11. Es pot observar ara que tots els no-importes han quedat assignats al valor "1".

Com s'ha esmentat abans, les expressions lògiques bàsiques són un pas cap a la implementació del circuit en portes lògiques. Els símbols normalitzats de les portes lògiques són els que es mostren en la Figura 2.1.

Segons el tipus d'expressió, l'estructura esquemàtica del circuit variarà. Quan l'expressió és una forma canònica, sempre es genera un circuit de màxim dos nivells, que vol dir que amb dos capes de portes la funció queda resolta. Si partim de la EC-I, l'estructura del circuit tindrà un primer nivell de portes AND i una porta OR amb la sortida (les portes inversores no es compten). Contràriament, si partim de la EC-II, l'estructura del circuit tindrà primer una capa de portes OR i una porta AND generant la sortida. Això es pot veure en el següent exemple.

Exemple 12 Dibuixar els esquemàtics dels circuits lògics que implementen les expressions canòniques de l'Exemple 11.

Si partim de l'EC-I obtindrem el circuit combinacional mostrat a la Figura 2.2. Com a cas particular es pot veure que la sortida w es resol amb una porta que la prenem del minterme m_7 de la funció z.

Si partim de l'EC-II obtindrem el circuit combinacional mostrat a la Figura 2.3. Com a cas particular es pot veure que la sortida z es resol amb una porta que la prenem del maxterme M_7 de la funció w.

Com es pot veure de l'Exemple 12 el nombre de termes suma o producte que intervenen en una expressió lògica té un efecte directe sobre la mida del circuit lògic, comptabilitzat amb nombre de portes lògiques o de transistors. És per aquest motiu que quan es vol sintetitzar un circuit combinacional, primer es mira de transformar l'expressió lògica a fi de reduir-ne el nombre de termes suma o producte.

En el següent apartat presentem un mètode de minimització d'expressions lògiques de dos nivells. El seu objectiu és reduir la mida de les expressions tot mantenint

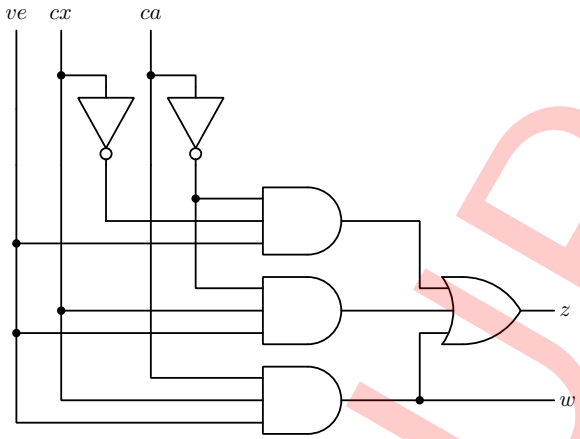


Figura 2.2: Implementació en portes lògiques de EC-I.

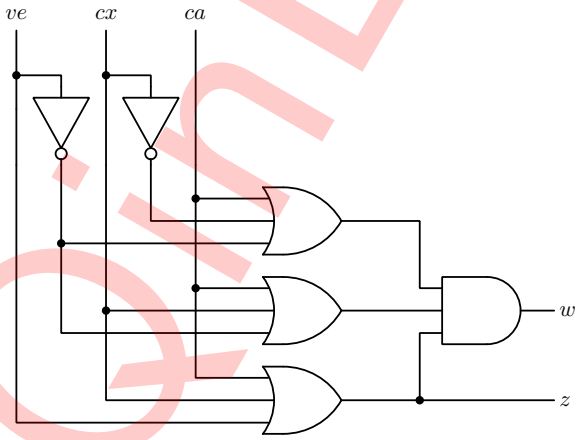


Figura 2.3: Implementació en portes lògiques de EC-II.

l'estructura SOP o POS. El resultat és que la seva implementació donarà lloc a circuits lògics amb dos nivells de portes però amb un nombre i mida de les portes menor que si es partís directament de les expressions canòniques EC-I o EC-II.

Capítol 3

Minimització d'Expressions Lògiques

L'objectiu de la minimització és identificar, en una expressió lògica, quins són els termes suma o producte redundants i eliminar-los, a fi d'obtenir una expressió final en dos nivells (SOP-min o POS-min) equivalent que tingui el nombre mínim de termes i literals. Vegeu-m'ho al següent exemple.

Exemple 13 Una funció lògica està especificada per les dues expressions lògiques SOP següents[1],

$$\begin{aligned} F(a,b,c,d)_{\text{EC-I}} =& \overline{a}\,\overline{b}\,\overline{c}\,\overline{d} + \overline{a}\,\overline{b}\,\overline{c}\,d + \overline{a}\,b\,c\,\overline{d} + \overline{a}\,b\,c\,d + a\,b\,c\,\overline{d} + a\,\overline{b}\,c\,\overline{d} + \\ & a\,\overline{b}\,c\,d + a\,b\,c\,d \\ F(a,b,c,d)_{\text{SOP-min}} =& \overline{a}\,\overline{b}\,\overline{c} + a\,c + b\,c \end{aligned}$$

Verifiquem la seva equivalència tot generant la TV a partir de cadascuna de les expressions. Representem l'esquema del circuit lògic i comptabilitzem els requeriments d'àrea derivats de cada implementació.

En la Taula 3.1 s'han avaluat les dues expressions i es pot observar que són equivalents, donen el mateix valor per qualsevol de les combinacions de les variables d'entrada.

La implementació circuital de l'expressió EC-I la trobem a la Figura 3.1. Es fa notar que el nombre de portes AND és igual al nombre de termes producte de l'expressió, i que el nombre d'entrades de cada AND es correspon amb el nombre de literals de cada terme producte. Alhora, el nombre d'entrades de la porta OR és igual al nombre de termes producte.

En una primera aproximació, l'avaluació dels requeriments d'àrea d'aquest circuit es pot fer comptabilitzant el nombre de portes lògiques i el seu nombre d'entrades. Cal tenir en compte que com més entrades tenen les portes lògiques més

[1] Quan no indueix a confusió, l'operador "·" es pot prendre de forma implícita, tal i com es mostra en les expressions EC-I i SOP-min.

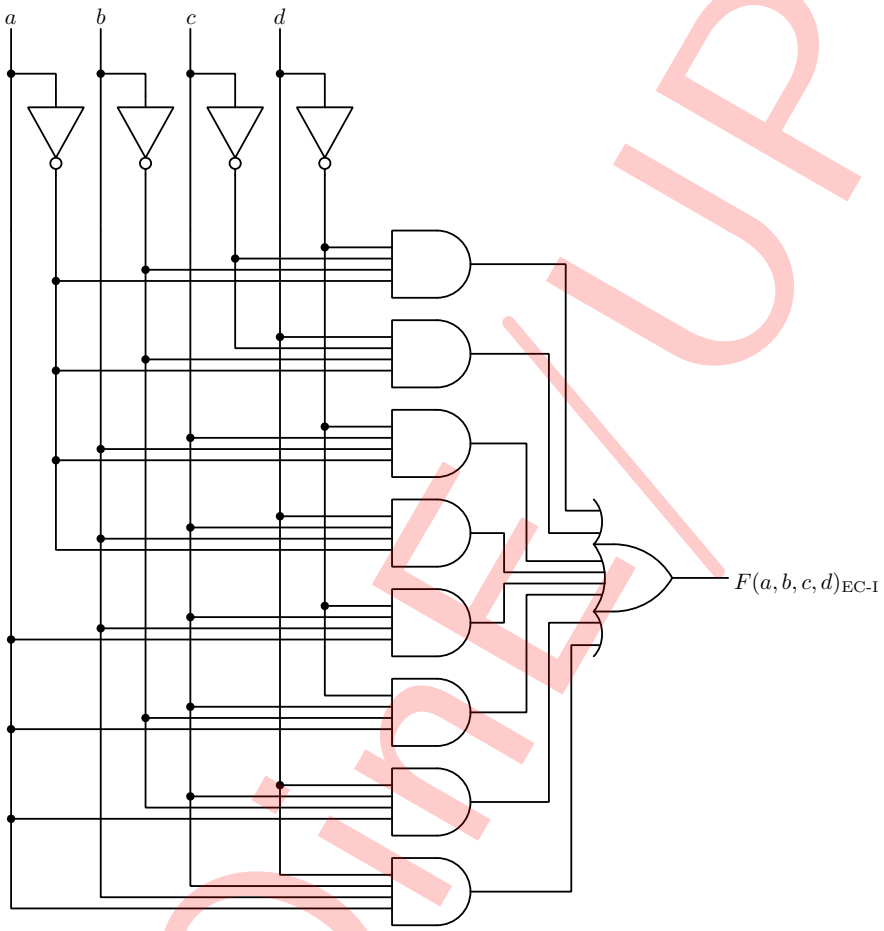


Figura 3.1: Implementació de l'expressió EC-I.

Taula 3.1: Taula de la veritat de les dues expressions.

a	b	c	d	$F(a,b,c,d)_{\text{EC-I}}$	$F(a,b,c,d)_{\text{SOP-min}}$
0	0	0	0	1	1
0	0	0	1	1	1
0	0	1	0	0	0
0	0	1	1	0	0
0	1	0	0	0	0
0	1	0	1	0	0
0	1	1	0	1	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	0	1	0	0
1	0	1	0	1	1
1	0	1	1	1	1
1	1	0	0	0	0
1	1	0	1	0	0
1	1	1	0	1	1
1	1	1	1	1	1

àrea necessiten, en una relació que és força lineal. En aquest cas, el nombre total de portes és de 4 × NOT + 8 × AND(4 entrades) + OR(8 entrades). Si multipliquem cada entrada per 2 tindrem una primera estimació del nombre de transistors requerits en una tecnologia CMOS[2]. Per tant, 2 × (4 + 8 × 4 + 8) = 88 transistors.

La implementació circuital de l'expressió SOP-min la trobem a la Figura 3.2. Es pot veure que el nombre de portes i entrades d'aquestes s'ha reduït substancialment. Fent l'estimació de l'àrea obtenim que el nombre de portes és de 3 × NOT + AND (3 entrades) + 2 × AND(2 entrades) + OR(3 entrades) que dona un total de transistors de 2 × (3 + 3 + 2 × 2 + 3) = 26 transistors, que és quasi la quarta part de l'anterior.

De l'Exemple 13 es pot veure que la minimització d'expressions lògiques dona com a resultat un estalvi d'àrea en la implementació circuital.

La minimització d'expressions lògiques es basa en el teorema de les adjacències que exposem seguidament.

[2]La tecnologia CMOS és la més emprada actualment en la implementació de circuits digitals [9] [2](“MOS Circuit / CMOS Logic”, pg. 234).

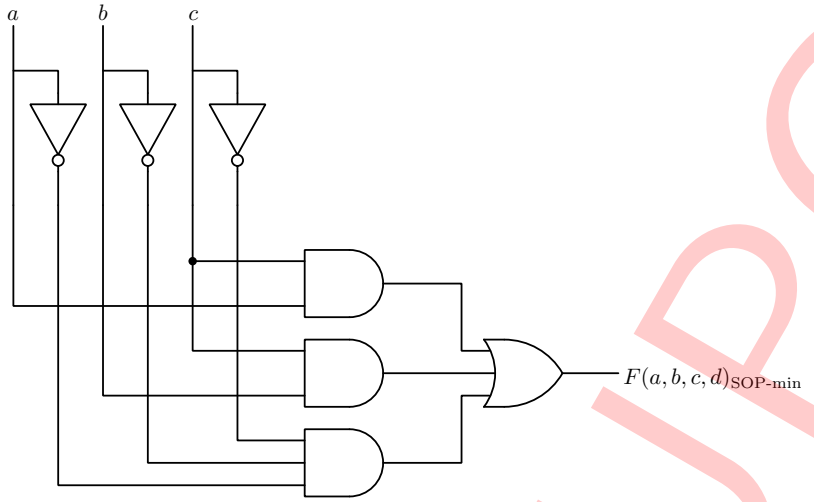


Figura 3.2: Implementació de l'expressió SOP-min.

3.1 Adjacència Lògica

Definició 1 (Adjacència en els termes producte) *Siguin dos termes producte p_1 i p_2. Són adjacents*[3] *si tots els seus literals són iguals excepte en un d'ells que aparegui negat i no-negat: $p_1 = p \cdot x$ i $p_2 = p \cdot \overline{x}$.*

Quan dos termes producte adjacents estan combinats amb una OR lògica, es poden simplificar al terme producte p aplicant els axiomes de l'àlgebra de commutació[4],

$$p \cdot x + p \cdot \overline{x} = p(x + \overline{x}) = p \cdot 1 = p \tag{3.1}$$

Definició 2 (Adjacència en els termes suma) *Siguin dos termes suma s_1 i s_2. Són adjacents*[3] *si tots els seus literals són iguals excepte en un d'ells que aparegui negat i no-negat: $s_1 = s + x$ i $s_2 = s + \overline{x}$.*

Quan dos termes suma adjacents estan combinats amb una AND lògica, es poden simplificar al terme suma s aplicant els axiomes de l'àlgebra de commutació[5],

$$(s + x)(s + \overline{x}) = s + x \cdot \overline{x} = s + 0 = s \tag{3.2}$$

La propietat de l'adjacència és el principi que s'aplica en la simplificació de les ELs tipus SOP i POS. Es van identificant aquestes i es van simplificant els termes produc-

[3] [2](“Minimal Forms / Algebraic Minimization”, pg. 282).

[4] Per ordre: propietats distributiva de la AND respecte de la OR, del complement en la OR i de la identitat en la AND.

[5] Per ordre: propietats distributiva de la OR respecte de la AND, del complement en la AND i de la identitat en la OR.

tes i sumes repetidament fins a no quedar cap nova adjacència. Veiem-ne un parell exemples.

Exemple 14 Detectem i simplifiquem totes les adjacències de l'expressió de tres variables EC-I en forma compacta $\sum(0,1,2,3,7)$.

Escrivim primer els mintermes de l'expressió en forma explícita,

$$\overline{a}\,\overline{b}\,\overline{c} + \overline{a}\,\overline{b}\,c + \overline{a}\,b\,\overline{c} + \overline{a}\,b\,c + a\,b\,c$$

Observant l'expressió, detectem les adjacències entre els mintermes següents: $\{m_0, m_1\}$, $\{m_2, m_3\}$ i $\{m_3, m_7\}$. Abans de simplificar-los hem de replicar el minterme m_3 en l'expressió[6] quedant,

$$\overline{a}\,\overline{b}\,\overline{c} + \overline{a}\,\overline{b}\,c + \overline{a}\,b\,\overline{c} + \overline{a}\,b\,c + \overline{a}\,b\,c + a\,b\,c$$

Ara simplifiquem les adjacències i obtenim,

$$\overline{a}\,\overline{b} + \overline{a}\,b + b\,c$$

que dona una reducció a tres termes producte. Si tornem a cercar adjacències en trobem una entre els termes producte $\{\overline{a}\overline{b}, \overline{a}b\}$. Simplificant de nou obtenim

$$\overline{a} + b\,c$$

que ja no presenta cap més adjacència i per tant, aquesta darrera expressió SOP és mínima.

Exemple 15 Detectem i simplifiquem totes les adjacències de l'expressió de tres variables EC-II en forma compacta $\prod(4,5,6)$.

Procedint de manera semblant a l'exemple anterior, escrivim primer els maxtermes de la funció de forma explícita,

$$(\overline{a} + b + c)(\overline{a} + b + \overline{c})(\overline{a} + \overline{b} + c)$$

Observant l'expressió, detectem les adjacències enter els maxtermes següents: $\{M_4, M_5\}$, $\{M_4, M_6\}$. Per tal de simplicar hem de duplicar el maxterme M_4 en l'expressió[6] quedant,

$$(\overline{a} + b + c)(\overline{a} + b + \overline{c})(\overline{a} + b + c)(\overline{a} + \overline{b} + c)$$

Simplifiquem les adjacències i obtenim l'expressió

$$(\overline{a} + b)(\overline{a} + c)$$

que ja no presenta més adjacències i per tant és una expressió mínima POS.

[6]Emprem la propietat de la idempotència que diu que $a + a = a$ i $a \cdot a = a$ [2](“Fundamental Properties / Useful Theorems”, pg. 172).

El procés de detecció d'adjacències i simplificació descrit en els Exemples 14 i 15 es formalitza en el teorema de la minimització.

Teorema 1 (De la minimització) *Sigui E una expressió lògica i x una variable lògica.*

$$E \cdot x + E \cdot \overline{x} = E \tag{3.3}$$

$$(E + x)(E + \overline{x}) = E \tag{3.4}$$

Tal com es pot veure en el Teorema de la minimització[7], la detecció o simplificació d'adjacències es farà sobre l'expressió SOP o POS donant lloc a les dues possibles expressions mínimes SOP-min o POS-min. Aquestes dues expressions no seran quasi mai iguals i per tant una de les dues pot ser més mínima que l'altra. Això ho podem veure en l'exemple següent.

Exemple 16 Sigui una funció lògica descrita per les expressions canòniques EC-I, $\sum(0, 1, 2, 3, 7)$, i EC-II, $\prod(4, 5, 6)$. Trobem l'expressió mínima.

Com hem vist en els Exemples 14 i 15 les expressions canòniques EC-I i EC-II tenen les expressions mínimes SOP-min, $\overline{a} + bc$, i POS-min, $(\overline{a} + b)(\overline{a} + c)$, respectivament. L'expressió mínima en dos nivells de la funció serà la més reduïda de les dues, SOP-min o POS-min. Hem d'avaluar doncs els requeriments d'àrees.

Per implementar la SOP-min necessitem: 1 NOT + 1 AND(2 entrades) + 1 OR(2 entrades) = 10 transistors. Per la POS-min necessitem: 1 NOT + 2 OR(2 entrades) + 1 AND(2 entrades) = 13 transistors. En conseqüència, l'expressió mínima és la SOP-min, $\overline{a} + b\,c$.

Com s'ha pogut veure en els exemples del 14 al 16 el procés de minimització d'una expressió lògica[8] segueix la metodologia indicada en la Figura 3.3. Es parteix de les expressions canòniques EC-I i EC-II i es detecten i simplifiquen totes les adjacències lògiques, procés que és iteratiu. De les expressions mínimes obtingudes, SOP-min i POS-min, es selecciona la més mínima. Quan per especificació es requereix d'una expressió explícitament SOP o POS, llavors el darrer pas s'obvia.

3.2 Mètode de Karnaugh o Mapa-K

Maurice Karnaugh va néixer el 1924 a la ciutat de Nova York. Va ser un matemàtic estatunidenc que va inventar el conegut mapa-K emprat en l'àlgebra de commutació. Aquesta tècnica la va proposar el 1954 mentre treballava als Laboratoris Bell [10]. El mètode de Karnaugh té com a objectiu la identificació sistemàtica de totes les adjacències lògiques d'una funció. Està especialment pensat per poder fer aquesta

[7] [2]("Minimal Forms / Algebraic Minimization", pg. 283).

[8] Sovint es parla de minimització d'una funció per indicar la minimització a dos nivells de la seva expressió lògica.

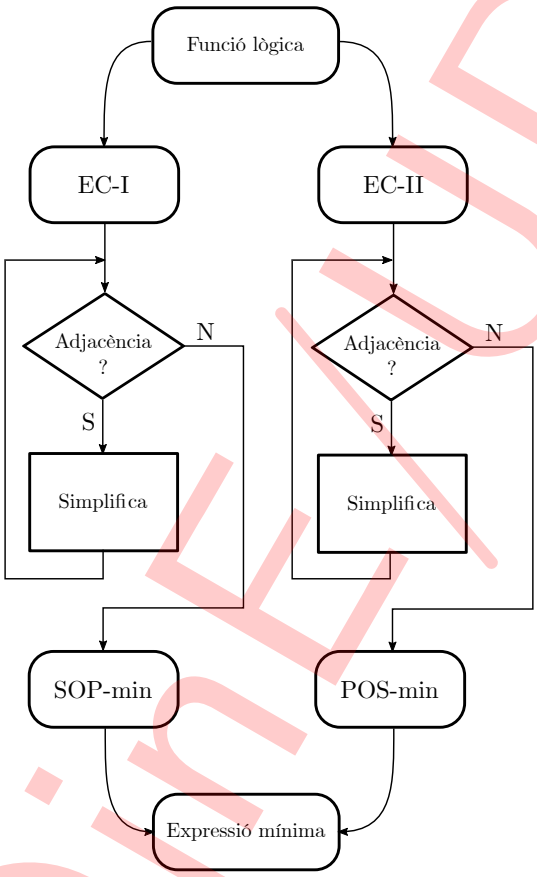


Figura 3.3: Metodologia de minimització de les expressions lògiques emprant el Teorema de la minimització.

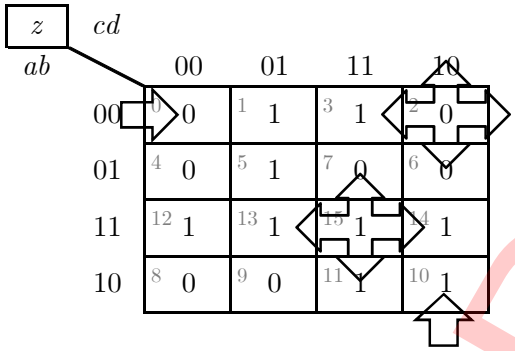


Figura 3.4: Mapa-K de la funció de quatre variables $Z(a,b,c,d) = \sum(1,3,5,10,11,12,13,14,15)$.

identificació de manera visual. Com a tècnica està limitada en el nombre de variables que pot manipular[9], habitualment no més de quatre variables, però és molt intuïtiu i no és iteratiu. Això vol dir que amb una única inspecció es poden detectar totes les adjacències de la funció.

Exemple 17 Representem el mapa-K de la funció de quatre variables especificada per l'expressió compacta $Z(a,b,c,d) = \sum(1,3,5,10,11,12,13,14,15)$.

El mapa el trobeu representat a la Figura 3.4.

Un mapa-K és una manera tabular d'especificar una funció lògica, semblant a les TVs, però amb unes propietats especials que s'indiquen en els punts següents:

1. La seva distribució és en forma matricial. Si el nombre de variables és parell es fa una distribució igual de les variables per files i columnes de manera que el mapa-K prengui una forma quadrada. Si el nombre de variables és senar, la distribució es fa de manera que la forma del mapa-K sigui la més quadrada possible.

2. La numeració de files i columnes correspon a l'avaluació de les variables d'entrada de la funció, en l'Exemple 17 per les variables $\{a,b,c,d\}$.

3. La numeració de les files i columnes es fa seguin una codificació binària reflectida, i.e. $\langle 00,01,11,10\rangle$, de manera que si ens movem una casella en horitzontal o en vertical només canviï de valor una variable.

   Per exemple, si en la Figura 3.4 ens movem de la casella 15 a la 14 l'única variable que canvia de valor és la $d:1\to 0$. O si ens movem de la casella 0 a la 4 canvia només la variable $b:0\to 1$.

[9] Existeixen altres mètodes més potents, com el tabular o el Quine-McCluskey, que són els emprats en el mètodes automàtics de minimització. Són mètodes iteratius i no intuïtius.

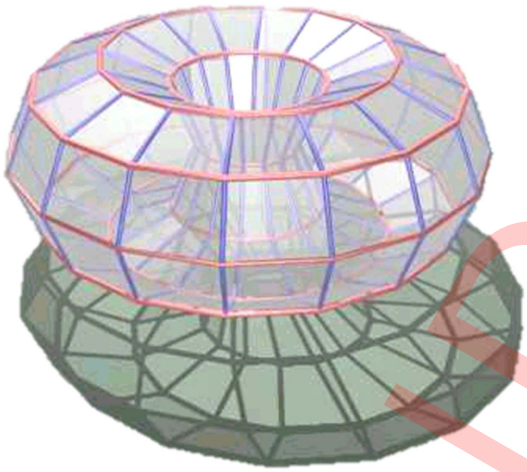


Figura 3.5: Continuïtat de les cel·les del mapa-K.

4. Com en una TV, en el mapa-K els 1ns representen els mintermes de la funció i els 0s els maxtermes. Si dos 1ns són contigus, en horitzontal o en vertical, els mintermes corresponents són adjacents. Si dos 0s són contigus, en horitzontal o en vertical, els maxtermes corresponents són adjacents.

   Per exemple, en el mapa-K anterior, els mintermes $m_{15} = a\,b\,c\,d$ i $m_{14} = a\,b\,c\,\overline{d}$ són adjacents ja que en el mapa-K són contigus. Ara be, els mintermes $m_{15} = abcd$ i el $m_3 = \overline{a}\,\overline{b}\,c\,d$ no ho són pas donat que en el mapa-K no són contigus.

   Semblantment, els maxtermes $M_0 = a{+}b{+}c{+}d$ i $M_4 = a{+}\overline{b}{+}c{+}d$ són adjacents ja que en el mapa-K els 0s són contigus i els maxtermes $M_0 = a + b + c + d$ i $M_2 = a + b + \overline{c} + d$ també ho són ja que els 0s són contigus pels límits exteriors del mapa-K.

5. El mapa-K té continuïtat pels seus extrems, és a dir, les caselles superiors estan connectades amb les inferiors i les de la dreta amb les de l'esquerra. A la Figura 3.5 es mostra una representació de la continuïtat de les cel·les en el mapa-K.

6. La continuïtat d'un nombre de 2^k per $k = 1, 2, ...$ de 1ns indica l'adjacència simultània de 2^k mintermes. De la mateixa manera, la continuïtat d'un nombre de 2^k per $k = 1, 2, ...$ de 0s indica l'adjacència simultània de 2^k maxtermes.

   Per exemple, en el mapa-K de la Figura 3.4 els mintermes m_{12}, m_{13}, m_{15} i m_{14} són adjacents degut a que formen un grup de quatre 1ns en fila. Semblantment, els mintermes m_{15}, m_{14}, m_{11} i m_{10} formen un agrupament de quatre 1ns contigus en dos columnes i dos files (formen un quadrat).

   Per la mateixa raó, els maxtermes M_0, M_2, M_4 i M_6 són adjacents perquè formen un agrupament de quatre 0s que són contigus en dos files i dos columnes (formant un quadrat).

Contràriament, els mintermes m_1, m_5 i m_{13} no són conjuntament adjacents ja que el nombre d'elements del grup (3) no es pot obtenir com a potència de 2. Una altra manera de veure-ho és intentar fer la simplificació dels termes producte i verificar que no es possible obtenir-ne un de sol. És a dir, amb $m_1 = \overline{a}\,\overline{b}\,\overline{c}\,d$, $m_5 = \overline{a}\,b\,\overline{c}\,d$ i $m_{13} = a\,b\,\overline{c}\,d$ podem inicialment simplificar les adjacències entre $p_{1,5} = m_1 + m_5 = \overline{a}\,\overline{c}\,d$ i $p_{5,13} = m_5 + m_{13} = b\,\overline{c}\,d$. A continuació, si observem els termes producte $p_{1,5}$ i $p_{5,13}$ veiem que no són adjacents ja que no comparteixen els mateixos literals, i per tant no ho podem arribar a simplificar a un únic terme producte.

7. L'extracció del terme producte simplificat d'un agrupament de 2^k 1ns és directa fent servir la següent metodologia. Ens movem dins de l'agrupament i identifiquem el valor que prenen les variables de la funció. Aquelles variables que canvien de valor les descartem del terme producte. Les variables que prenen valor "1" les incloem directament i les que prenen valor "0" les incloem invertides.

   Per exemple, en l'agrupament de quatre 1ns corresponent als mintermes m_{15}, m_{14}, m_{11} i m_{10}, el terme producte simplificat serà $p_{10,11,14,15} = m_{10} + m_{11} + m_{14} + m_{15} = ac$, ja que $a = 1$, $c = 1$ i b i d canvien de valor.

8. L'extracció del terme suma simplificat d'un agrupament de 2^k 0s és directa fent servir la següent metodologia. Ens movem dins de l'agrupament i identifiquem el valor que prenen les variables de la funció. Aquelles variables que canvien de valor les descartem del terme suma. Les variables que prenen valor "1" les incloem invertides i les que prenen "0" ho fem directament.

   Per exemple, en l'agrupament de quatre 0s corresponent als maxtermes M_0, M_2, M_4 i M_6, el terme suma simplificat serà $s_{0,2,4,6} = M_0 \cdot M_2 \cdot M_4 \cdot M_6 = a + d$, ja que $a = 0$, $d = 0$ i b i c canvien de valor.

9. Qualsevol agrupament de 2^k 1ns o 0s dona lloc a un terme producte o suma amb un nombre de literals igual a $n - k$ on n és el nombre de variables de la funció.

3.2.1 Implicants i Implicats d'una Funció

Definició 3 (Implicant d'una funció) *Un implicant*[10] *d'una funció F és un terme producte p que per aquelles combinacions A de les variables d'entrada que val "1" la funció dona "1".*

$$p(A) = 1 \Rightarrow F(A) = 1$$

També podem dir que el conjunt de 1ns del terme producte pertany al conjunt de 1ns de la funció,

$$\{p\}_1 \in \mathcal{F}_1$$

[10] [2]("Minimal Forms / Prime Implicants & ss.", pg. 287)

Observació 1 (Detecció d'un implicant en el mapa-K) *En un mapa-K, un implicant ve representat per un agrupament de 2^k 1ns.*

Definició 4 (Implicat d'una funció) *Un implicat*[10] *d'una funció F és un terme suma s que per aquelles combinacions A de les variables d'entrada que val "0" la funció dona "0".*

$$s(A) = 0 \Rightarrow F(A) = 0$$

També podem dir que el conjunt de 0s del terme suma pertany al conjunt de 0s de la funció,

$$\{s\}_0 \in \mathcal{F}_0$$

Observació 2 (Detecció d'un implicat en el mapa-K) *En un mapa-K, un implicat ve representat per un agrupament de 2^k 0s.*

Exemple 18 Detectem tots els implicants de la funció $Z(a, b, c, d)$ de l'Exemple 17 a partir de l'observació del mapa-K de la Figura 3.4.

Per observació veiem que es poden fer agrupacions de dos i quatre 1ns. En la Taula 3.2 s'indiquen totes les agrupacions possibles i l'implicants associats.

Taula 3.2: Tots els implicants de la funció $Z(a, b, c, d)$ de la Figura 3.4.

Implicants de la funció $Z(a, b, c, d)$			
Agrupacions de dos 1ns			
$p_{1,3}$	$\overline{a}\,\overline{b}\,d$	$p_{14,12}$	$a\,b\,\overline{d}$
$p_{1,5}$	$\overline{a}\,\overline{c}\,d$	$p_{15,11}$	$a\,c\,d$
$p_{5,13}$	$b\,\overline{c}\,d$	$p_{14,10}$	$a\,c\,\overline{d}$
$p_{12,13}$	$a\,b\,\overline{c}$	$p_{11,10}$	$a\,\overline{b}\,c$
$p_{13,15}$	$a\,b\,d$	$p_{11,3}$	$\overline{b}\,c\,d$
$p_{15,14}$	$a\,b\,c$		
Agrupacions de quatre 1ns			
$p_{12,13,15,14}$	$a\,b$	$p_{15,14,11,10}$	$a\,c$

Exemple 19 Detectem tots els implicats de la funció $Z(a, b, c, d)$ de l'Exemple 17 a partir de l'observació del mapa-K de la Figura 3.4.

Per observació veiem que es poden fer agrupacions de dos i quatre 0s. En la Taula 3.3 s'indiquen totes les agrupacions possibles i l'implicats associats.

Taula 3.3: Tots els implicats de la funció $Z(a,b,c,d)$ de la Figura 3.4.

Implicats de la funció $Z(a,b,c,d)$			
Agrupacions de dos 0s			
$s_{0,2}$	$a+b+d$	$s_{6,4}$	$\overline{a}+b+\overline{d}$
$s_{0,4}$	$a+c+d$	$s_{8,9}$	$\overline{a}+b+c$
$s_{2,6}$	$a+\overline{c}+d$	$s_{8,0}$	$b+c+d$
$s_{7,6}$	$a+\overline{b}+\overline{c}$		
Agrupacions de quatre 0s			
$s_{0,2,4,6}$	$a+d$		

3.2.2 Implicants i Implicats Primers d'una Funció

Definició 5 (Implicant primer d'una funció) *Un terme producte p d'una funció F és primer (p^*), si quan li eliminem un literal deixa de ser implicant de la funció.*

$$p^* = \prod_{i=0}^{k} \dot{x}_i \Rightarrow \left\{ \prod_{i=0}^{k-1} \dot{x}_i \right\}_1 \notin \mathcal{F}_1$$

Observació 3 (Detecció d'un implicant primer en el mapa-K) *En un mapa-K, un implicant primer ve representat per un agrupament de 2^k 1ns de mida màxima.*

Definició 6 (Implicat primer d'una funció) *Un terme suma s d'una funció F és primer (s^*), si quan li eliminem un literal deixa de ser implicat de la funció.*

$$s^* = \sum_{i=0}^{k} \dot{x}_i \Rightarrow \left\{ \sum_{i=0}^{k-1} \dot{x}_i \right\}_0 \notin \mathcal{F}_0$$

Observació 4 (Detecció d'un implicat primer en el mapa-K) *En un mapa-K, un implicat primer ve representat per un agrupament de 2^k 0s de mida màxima.*

Es fa notar que en el mapa-K, un agrupament de mida màxima pot fer intersecció amb altres agrupament però mai quedar circumscrit completament dins d'un altre. En aquest darrer cas, l'agrupament no seria de mida màxima. En el següent exemple s'il·lustra això.

Exemple 20 En el mapa-K de la Figura 3.4 indiquem tots els implicants i implicats primers de la funció $Z(a,b,c,d)$.

En la Figura 3.6 es mostren els agrupaments de 1ns corresponents als implicants primers de la funció. En la Taula 3.2 es mostren els termes productes associats.

En la Figura 3.7 es mostren els agrupaments de 0s corresponents als implicats primers de la funció. En la Taula 3.3 es mostren els termes suma associats.

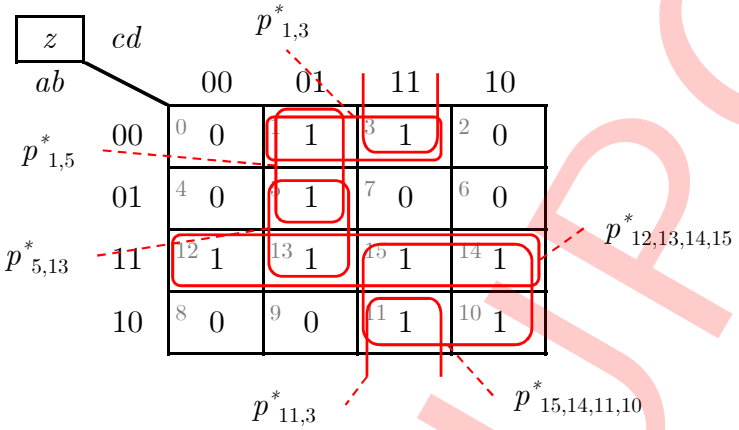


Figura 3.6: Implicants primers representats en el mapa-K.

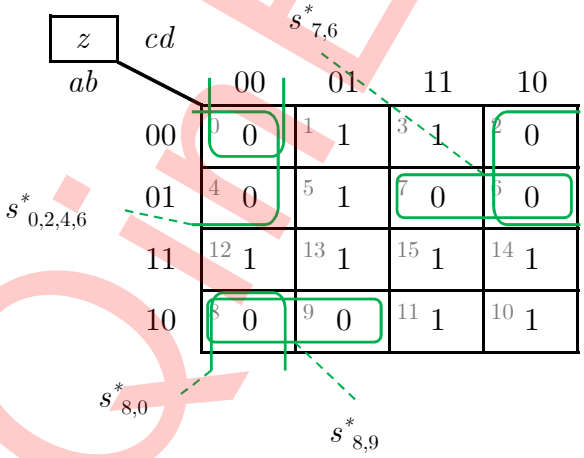


Figura 3.7: Implicats primers representats en el mapa-K.

3.2.3 Metodologia de Minimització amb el mapa-K

La metodologia de minimització[11] amb els mapes-K consisteix en detectar els implicants/implicats (termes) primers de la funció i fer una selecció mínima d'aquests per tal de cobrir tots els 1ns/0s de la funció. Com és de suposar, no tots els termes primers de la funció són necessaris per fer aquesta cobertura, altrament, si incloem en l'expressió mínima termes primers redundants estarem generant expressions més complexes del necessari, i per tant no seran mínimes.

Per facilitar aquesta tria, els termes primers es classifiquen en tres categories: els essencials, els no essencials i els absolutament no essencials.

Definició 7 (Implicant primer essencial) *Sigui un implicant primer p^* d'una funció F. Aquest és essencial p^{*e} si en el mapa-K, algun dels 1ns del seu agrupament no està cobert per cap altre implicant primer.*

En el mapa-K de la Figura 3.6 són essencials els implicants primers $p^{*e}_{12,13,14,15}$ ja que el 1_{12} no està cobert per cap altre p^*_i i el $p^{*e}_{15,14,11,10}$ degut a que el 1_{10} no està cobert per cap altre p^*_i.

Definició 8 (Implicat primer essencial) *Sigui un implicat primer s^* d'una funció F. Aquest és essencial s^{*e} si en el mapa-K, algun dels 0s del seu agrupament no està cobert per cap altre implicat primer.*

En el mapa-K de la Figura 3.7 són essencials els implicats primers $s^{*e}_{8,9}$, $s^{*e}_{0,2,4,6}$ i $s^{*e}_{7,6}$ ja que els 0_9, 0_4, 0_2 i 0_7 no estan coberts per cap altre s^*_i.

Definició 9 (Implicant primer no essencial) *Sigui un implicant primer p^* d'una funció F. Aquest és no essencial (p^{*n}) si en el mapa-K tots els seus 1ns estan coberts per altres implicants primers i almenys un dels 1ns està cobert per un altre implicant primer no essencial.*

En el mapa-K de la Figura 3.6 són no essencials els implicants primers $p^{*n}_{5,3}$, $p^{*n}_{1,5}$, $p^{*n}_{1,3}$ i $p^{*n}_{11,3}$. En el $p^{*n}_{5,3}$ el 1_{13} està cobert per un implicant primer essencial ($p^{*n}_{12,13,14,15}$) i el 1_5 per un altre implicant primer no essencial ($p^{*n}_{1,5}$). Els 1ns del $p^{*n}_{1,5}$ estan coberts per altres implicants primers no essencials ($p^{*n}_{5,13}$ i $p^{*n}_{1,3}$). Semblantment li passa amb els 1ns del $p^{*n}_{1,3}$ i finalment a l'implicant primer no essencial $p^{*n}_{11,3}$ un 1 està cobert per un implicant primer essencial ($p^{*e}_{15,14,11,10}$) i l'altre 1 per $p^{*n}_{1,3}$.

Definició 10 (Implicat primer no essencial) *Sigui un implicat primer s^* d'una funció F. Aquest és no essencial (s^{*n}) si en el mapa-K tots els seus 0s estan coberts per altres implicats primers i almenys un dels 0s està cobert per un altre implicat primer no essencial.*

[11] [2]("The K-map Method / Minimization", pg 292).

En el mapa-K de la Figura 3.7 no hi ha cap implicat primer no essencial. En el cas de $s^*_{8,0}$ té els seus 0s coberts per dos implicats primers essencials i per tant no pertany a aquesta categoria ja que no té com a mínim un 0 cobert per un implicat primer no essencial.

Definició 11 (Implicant primer absolutament no essencial) *Sigui un implicant primer p^* d'una funció F. Aquest és absolutament no essencial (p^{*a}) si en el mapa-K tots els seus 1ns estan coberts per altres implicants primers essencials.*

En el mapa-K de la Figura 3.6 no hi ha cap implicant primer absolutament no essencial.

Definició 12 (Implicat primer absolutament no essencial) *Sigui un implicat primer s^* d'una funció F. Aquest és absolutament no essencial (s^{*a}) si en el mapa-K tots els seus 0s estan coberts per altres implicats primers essencials.*

En el mapa-K de la Figura 3.7 ho és el $s^{*a}_{8,0}$ degut a que els seus dos 0s estan coberts també pels implicats primers essencials, $s^{*e}_{8,9}$ i $s^{*e}_{0,2,4,6}$.

Així doncs, la metodologia de minimització emprant els mapes-K es sistematitza de la següent manera:

3.2.4 Cerca de l'Expressió SOP-min/POS-min d'una Funció Lògica

1. Representar el mapa-K de la funció lògica.
2. Identificar els implicants primers essencials (p^{*e}) i no essencials (p^{*n}).
3. Seleccionar tots els implicants primers essencials (p^{*e}).
4. Per la resta de 1ns no coberts, fer una tria mínima d'implicants primers no-essencials (p^{*n}) que els cobreixin tots.
5. Extreure els termes producte de p^{*e}_i i p^{*n}_j.
6. Fer una OR de tots els termes productes per generar l'expressió SOP-min.

Exemple 21 Extraiem totes les expressions SOP-min de la funció lògica representada en el mapa-K de la Figura 3.6.

En la Figura 3.8 es presenta la selecció del implicants primers essencials i una tria dels no essencials que permeten fer una cobertura mínima dels 1ns de la funció.

Els termes producte associats són: $p^{*e}_{12,13,14,15} = a\,b$, $p^{*e}_{15,14,11,10} = a\,c$, $p^{*n}_{1,5} = \overline{a}\,\overline{c}\,d$ i $p^{*n}_{11,3} = \overline{b}\,c\,d$. Fent la OR obtenim l'expressió SOP-min,

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

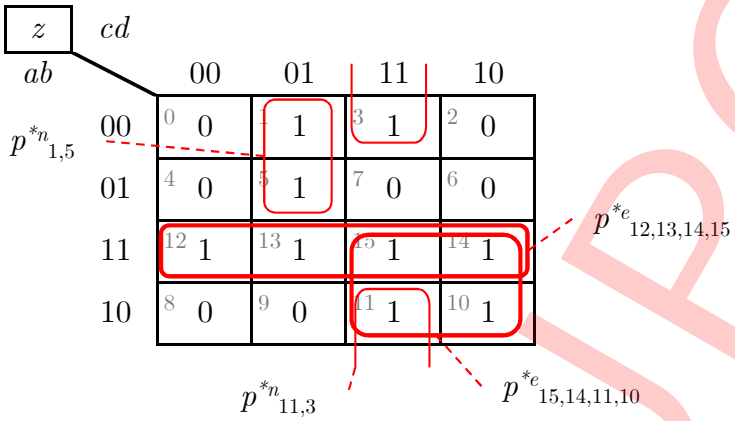


Figura 3.8: Cobertura dels 1ns de la funció fent una selecció de tots els implicants primers essencials i alguns no essencials.

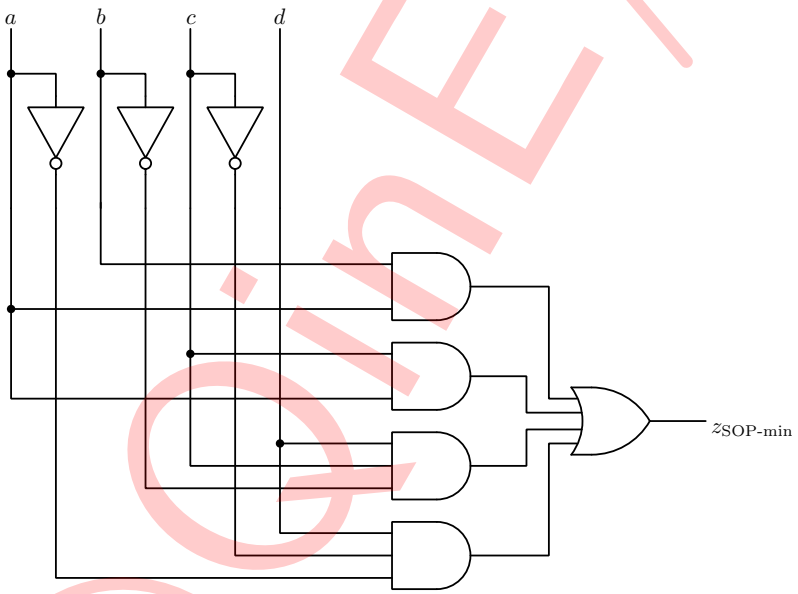


Figura 3.9: Implementació de l'expressió SOP-min ($z = a\,b + a\,c + \overline{a}\,\overline{c}\,d + \overline{b}\,c\,d$).

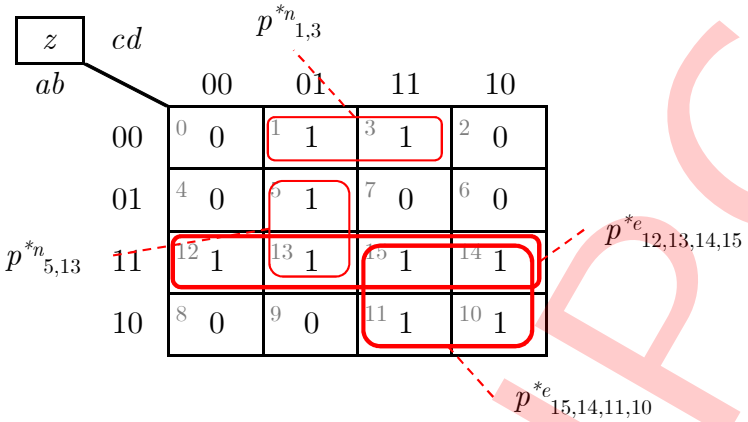


Figura 3.10: Cobertura alternativa dels 1ns de la funció fent una selecció de tots els implicants primers essencials i alguns no essencials.

En la Figura 3.9 es representa el circuit lògic. Emprant el mateix càlcul de l'Exemple 16, el nombre total de transistors requerits per aquesta implementació és de 34.

Una altre alternativa de selecció dels implicants primers no essencials és la que es mostra a la Figura 3.10. En aquest segon cas els productes associats són: $p^{*e}_{12,13,14,15} = a\,b$, $p^{*e}_{15,14,11,10} = a\,c$, $p^{*n}_{5,13} = b\,\overline{c}\,d$ i $p^{*n}_{1,3} = \overline{a}\,\overline{b}\,d$. Fent la OR obtenim l'expressió SOP-min alternativa,

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

En la Figura 3.11 es representa el circuit lògic. L'estimació del nombre total de transistors de la implementació és de 34, igual que l'anterior.

Com s'ha pogut veure en l'exemple anterior, a diferència de les expressions canòniques, les expressions mínimes d'una funció lògica no són úniques. En concret, no ho són quan en el mapa-K hi ha implicants primers no essencials. D'aquí doncs en podem extreure la següent observació.

Observació 5 *Les expressions mínimes SOP-min i POS-min d'una funció lògica no són sempre úniques. Només ho seran quan en el mapa-K la cobertura de 1ns/0s es faci únicament amb implicants/implicats primers essencials.*

El cas d'una expressió mínima única es veu en el següent exemple.

Exemple 22 Extraiem l'expressió POS-min de la funció lògica representada en el mapa-K de la Figura 3.7.

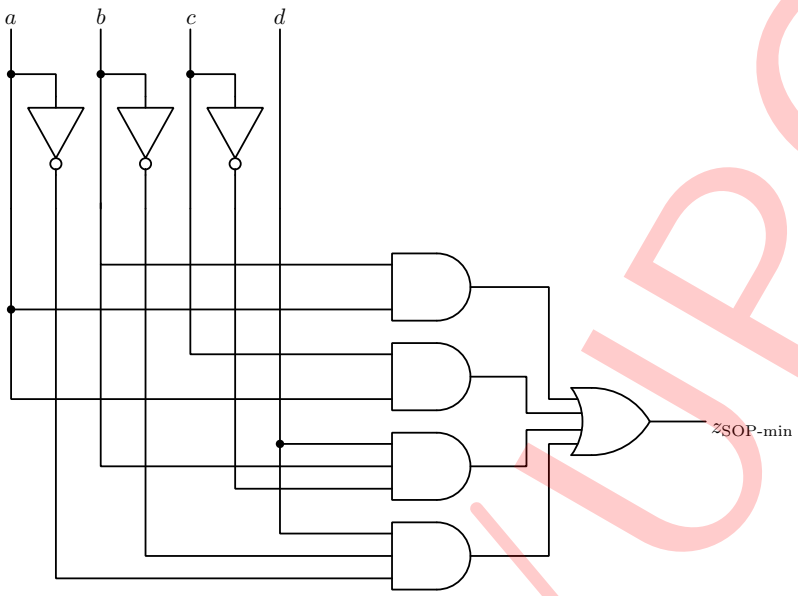


Figura 3.11: Implementació de l'expressió SOP-min ($z = a\,b + a\,c + b\,\bar{c}\,d + \bar{a}\,\bar{b}\,d$).

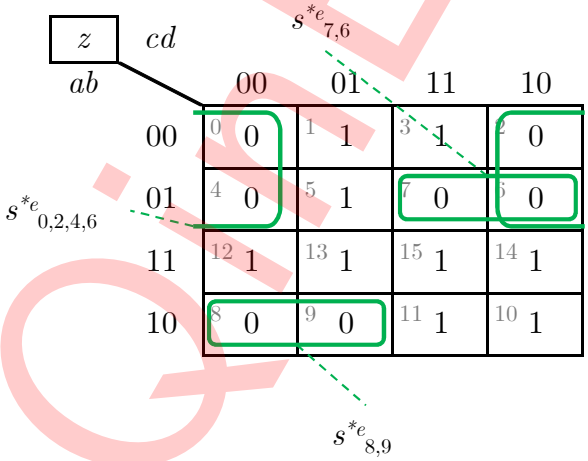


Figura 3.12: Cobertura dels 0s de la funció amb els implicats primers essencials.

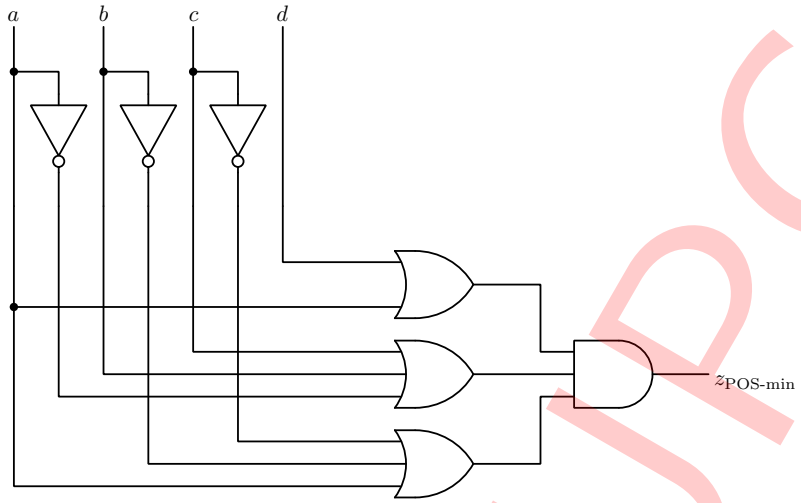


Figura 3.13: Implementació de l'expressió POS-min $(z = (a + \overline{b} + \overline{c})(\overline{a} + b + c)(a + d))$.

En la Figura 3.12 es mostra la cobertura de 0s que s'assoleix amb els implicats primers essencials $s^{*e}_{7,6} = a + \overline{b} + \overline{c}$, $s^{*e}_{8,9} = \overline{a} + b + c$, $s^{*e}_{0,2,4,6} = a + d$. Fent la AND obtenim l'expressió POS-min que en aquest cas és única,

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

En la Figura 3.13 es mostra el circuit lògic. Estimant l'àrea de la implementació ens dóna un total de 28 transistors que és menor que en la versió SOP-min de l'expressió. En aquest cas l'elecció de l'expressió mínima seria la POS-min, segons la metodologia indicada en la Figura 3.3.

3.3 Minimització de Funcions Parcialment Especificades

Quan una funció està parcialment especificada (presenta no-importes a la TV) la minimització de l'EL amb el mapa-K resulta especialment efectiva. Les X del mapa-K les podem usar com a comodins a fi de maximizar les agrupacions[12] de 1ns o 0s. Això ho veiem en l'exemple que es presenta a continuació.

Exemple 23 Es defineix una funció lògica $F(a, b, c, d)$ especificada per la TV de la Taula 3.4. Determinem l'expressió mínima SOP-min/POS-min en els tres supòsits següent: assignació de tots els X a "1", assignació de tots els X a "0" i assignació òptima dels X.

[12] [2]("Larger Problems / Minimization with Don't-Cares", pg. 309).

Taula 3.4: Taula de la veritat de la funció de quatre variables $F(a, b, c, d)$.

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

Assignació de tots els X a "1"

Per tal d'obtenir l'expressió SOP-min construim el mapa-K amb les X = 1, que es mostra a la Figura 3.14. En aquests hi ha indicats els implicants primers que fan una cobertura mínima de tots els 1ns de la funció. Tots els implicants primers són essencials i els seus termes productes són: $p^{*e}_{2,3} = \overline{a}\,\overline{b}\,c$, $p^{*e}_{8,9,12,13} = a\,\overline{c}$ i $p^{*e}_{1,3,5,7,11,13,15} = d$. Per tant l'expressió mínima serà,

$$f_{\text{SOP-min}} = \overline{a}\,\overline{b}\,c + a\,\overline{c} + d$$

El nombre estimat de transistors per a aquesta implementació és de 22.

Per la POS-min cerquem els implicats primers que assoleixen una cobertura mínima de 0s. En la Figura 3.15 podem veure el mapa-K amb aquesta cobertura indicada. Trobem dos implicats primers essencials, $s^{*e}_{0,4} = a + c + d$ i $s^{*e}_{10,14} = \overline{a} + \overline{c} + d$ i un de no essencials, $s^{*n}_{6,14} = \overline{b} + \overline{c} + d$. Fent la AND d'aquests obtenim l'expressió minima,

$$f_{\text{POS-min}} = (a + c + d)(\overline{a} + \overline{c} + d)(\overline{b} + \overline{c} + d)$$

Que es pot implementar amb 30 transistors. Es conclou doncs que amb l'assignació dels no-importa a "1" l'expressió mínima es pot implementar amb 22 transistors a partir de la SOP-min.

Assignació de tots els X a "0"

Ara recalculem l'expressió mínima canviant l'assignació X = 0. De nou cerquem la SOP-min fent una cobertura mínima de 1ns amb implicants primers. En la Figura 3.16 trobem el mapa-K d'aquest cas amb dos implicants primers essencials,

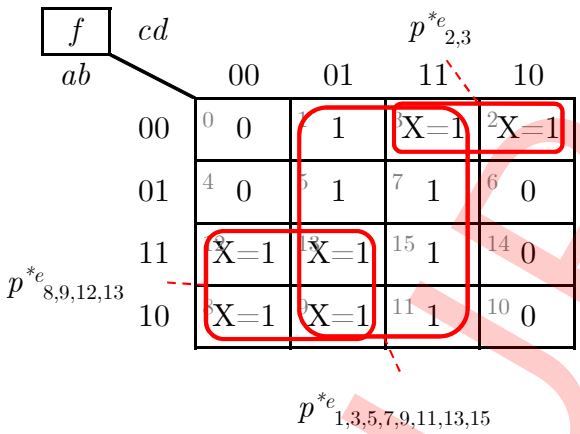


Figura 3.14: Cobertura dels 1ns de la funció assumint X = 1.

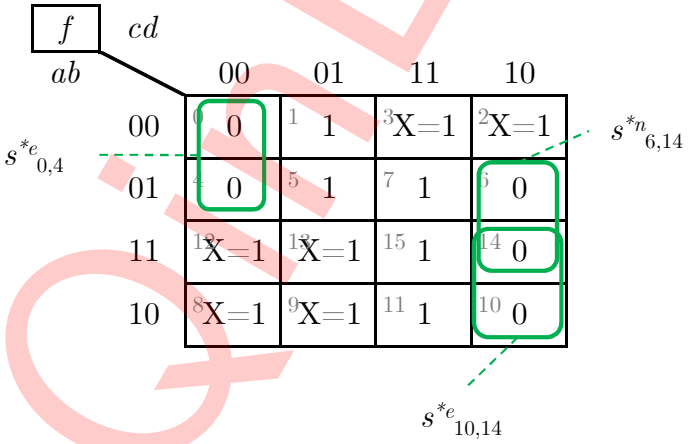


Figura 3.15: Cobertura dels 0s de la funció assumint X = 1.

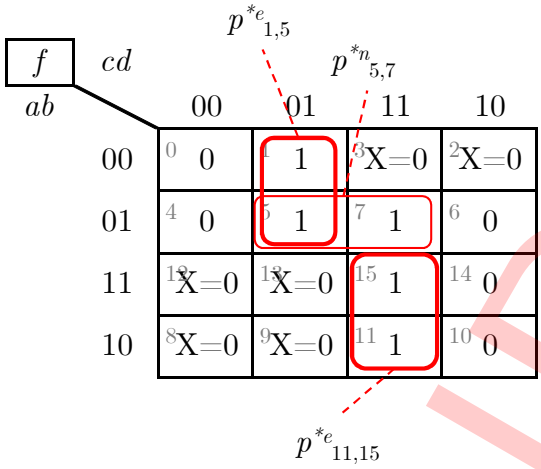


Figura 3.16: Cobertura dels 1ns de la funció assumint X = 0.

$p^{*e}_{1,5} = \overline{a}\,\overline{c}\,d$ i $p^{*e}_{11,15} = a\,c\,d$, i un implicant primer no essencial $p^{*n}_{5,7} = \overline{a}\,b\,d$. Fent la OR obtenim l'expressió mínima,

$$f_{\text{SOP-min}} = \overline{a}\,\overline{c}\,d + a\,c\,d + \overline{a}\,b\,d$$

que es pot implementar amb 28 transistors.

Cerquem ara l'expressió POS-min fent la cobertura mínima de 0s amb implicats primers de la funció. A la Figura 3.17 hi ha el mapa-K. Podem identificar tres implicats primers essencials que són: $s^{*e}_{0,2,4,6,8,10,12,14} = d$, $s^{*e}_{2,3} = a + b + \overline{c}$ i $s^{*e}_{8,9,12,13} = \overline{a} + c$. Fent la AND obtenim l'expressió POS-min,

$$f_{\text{POS-min}} = (a + b + \overline{c})(\overline{a} + c)d$$

que es pot implementar amb 20 transistors. Es conclou doncs que l'expressió POS-min amb una assignació de X = 0 és la que dona una implementació més mínima de la funció de la Taula 3.4.

Assignació òptima dels X

L'assignació òptima de les X es fa durant la cerca de la cobertura mínima de 1ns o 0s. Anem primer a minimitzar l'expressió SOP i per tant a fer una cobertura de 1ns. Assignarem X = 1 només en aquelles posicions que ens permetin engrandir els implicants primers (encerclaments de 1ns) i la resta els assignarem a X = 0.

En la Figura 3.18 es mostra aquesta assignació òptima. L'implicant primer essencial $p^{*e}_{1,3,5,7,9,11,13,15} = d$ té una mida màxima gràcies a l'assignació a "1" dels seus no importes. Amb aquest tots els 1ns de la funció queden coberts i conseqüentment la resta les deixem a X = 0. Per tant, l'expressió mínima només té un implicant primer i és,

$$f_{\text{SOP-min}} = d$$

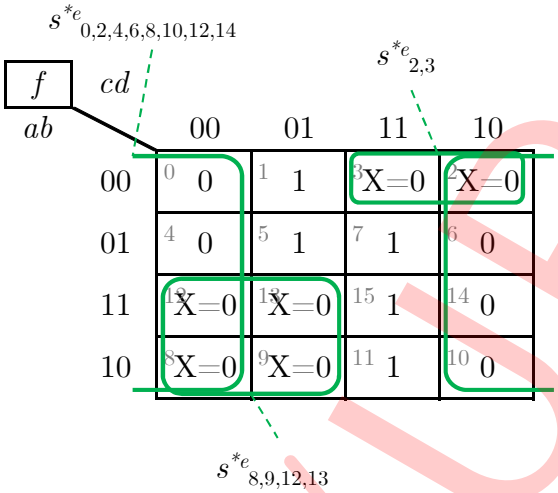


Figura 3.17: Cobertura dels 0s de la funció assumint X = 0.

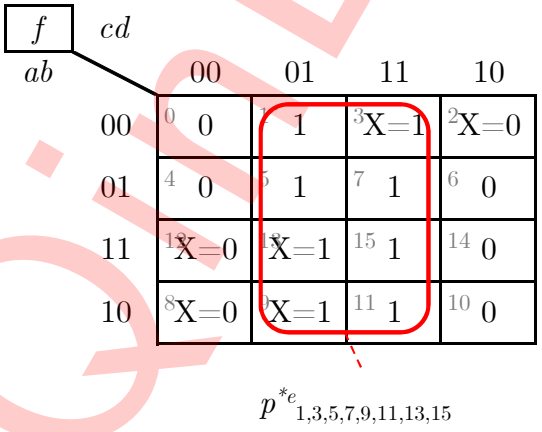


Figura 3.18: Assignació òptima de X = 1 de la funció.

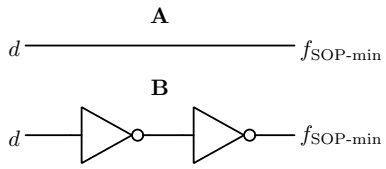


Figura 3.19: Implementació de l'expressió POS-min $f_{\text{SOP-min}} = d$. **A** directa i **B** regenerant el senyal.

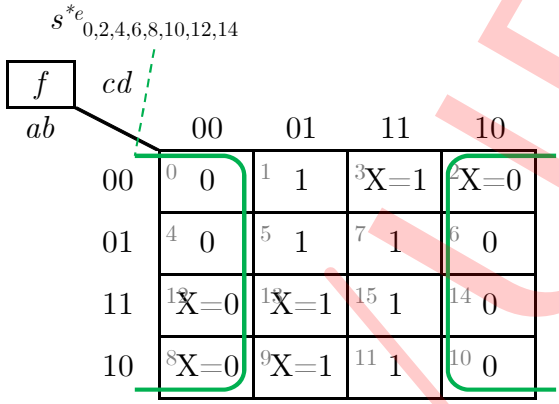


Figura 3.20: Assignació òptima de X = 1 de la funció.

que no necessita cap transistor per ser implementada o com a màxim 4 per tal de garantir la regeneració del senyal[13]. Això ho podem veure en la implementació circuital de la Figura 3.19.

Quan minimitzem l'expressió POS, procedirem de manera semblant en cercar la cobertura de 0s. Identifiquem els implicats primers de la funció i posteriorment assignem les X = 0 de manera que puguem augmentar la mida dels implicats al màxim. La resta els deixem amb l'assignació X = 1. El resultat es pot veure a la Figura 3.20. Identifiquem un únic implicat primer essencial, el $s^{*e}_{0,2,4,6,8,10,12,14} = d$ amb el qual cobrim tots els 0s de la funció. En conseqüència l'expressió POS-min serà,

$$f_{\text{POS-min}} = d$$

que dona la casualitat que és igual que la SOP-min però que en general no té perquè ser així.

Del que s'ha vist en l'exemple anterior se'n pot extreure la següent observació pel que fa a la assignació òptima de les X en el procés de minimització pel mapa-K.

[13] En la tecnologia CMOS les portes lògiques, a banda de les operacions lògiques, també regeneren els nivells de tensió dels senyals lògics. És aquest motiu el que fa la tecnologia CMOS tan robusta i mundialment emprada [9].

Observació 6 (Assignació òptima de les X en SOP-min) *Sigui el mapa-K d'una funció lògica parcialment especificada. Identifiquem inicialment els implicants primers essencials i no essencials. Seguidament assignem les* $\mathrm{X} = 1$ *de la funció de tal manera que augmentem al màxim la mida dels agrupaments associats als esmentats implicants primers. La resta les deixem a* $\mathrm{X} = 0$*. Si en aquest procés alguns dels implicants primers passen a ser absolutament no essencials els eliminem del mapa-K. Amb els implicants primers resultants cerquem una cobertura mínima dels 1ns de la funció i fent una OR d'aquests generem l'expressió SOP-min.*

Semblantment per la minimització de les expressions POS podem fer l'observació següent.

Observació 7 (Assignació òptima de les X en POS-min) *Sigui el mapa-K d'una funció lògica parcialment especificada. Identifiquem inicialment els implicats primers essencials i no essencials. Seguidament assignem les* $\mathrm{X} = 0$ *de la funció de tal manera que augmentem al màxim la mida dels agrupaments associats als esmentats implicats primers. La resta les deixem a* $\mathrm{X} = 1$*. Si en aquest procés alguns dels implicats primers passen a ser absolutament no essencials els eliminem del mapa-K. Amb els implicats primers resultats cerquem una cobertura mínima dels 0s de la funció i fent una AND d'aquests generem l'expressió POS-min.*

Veiem-ne un altre exemple.

Exemple 24 Es disposa de la funció $F(a, b, c)$ especificada per la TV de la Figura 3.5. Anem, en primer lloc, a obtenir l'expressió mínima SOP. Per això

Taula 3.5: Funció de tres variables $F(a, b, c)$.

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

identifiquem inicialment els implicants primers essencials i no essencials de la funció en el mapa-K, vegeu-ho en la Figura 3.21.

Identifiquem dos implicants primers essencials, $p^{*e}_{1,3} = \overline{b}\,c$ i $p^{*e}_{5,7} = a\,c$. Cap d'aquests dos agrupaments pot ser ampliat fent una assignació de $\mathrm{X} = 1$ a l'únic no-importa de la funció i per tant l'expressió SOP mínima és,

$$f_{\text{SOP-min}} = \overline{b}\,c + a\,c$$

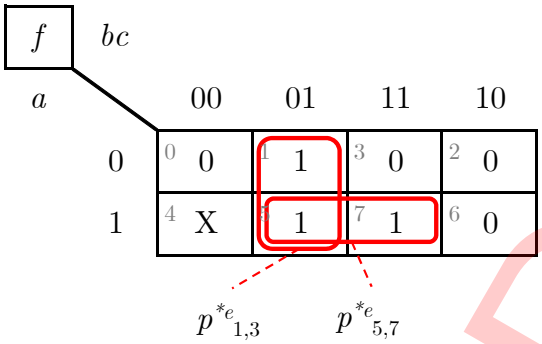


Figura 3.21: Implicants primers de la funció.

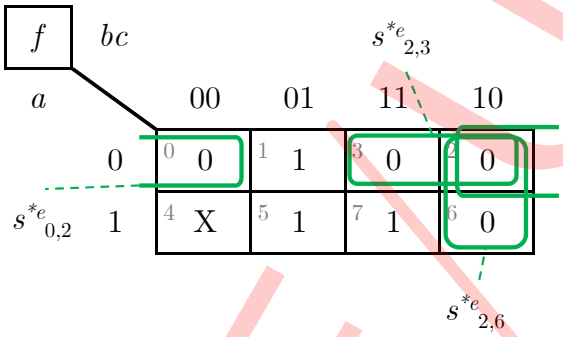


Figura 3.22: Implicants primers de la funció.

Anem a extreure ara l'expressió POS-min. Fem de nou la identificació dels implicats primers essencials i no essencials de la funció, es pot veure en la Figura 3.22.

Trobem tres implicats primers essencials, $s^{*e}_{0,2}$, $s^{*e}_{2,3}$, $s^{*e}_{2,6}$. Es pot veure que si fem X = 0 l'agrupament de 0s de l'implicat primer $s^{*e}_{2,6}$ pot augmentar de dos a quatre 0s passant a ser l'implicat primer $s^{*e}_{0,2,4,6}$. Com a conseqüència d'això l'agrupament de 0s de l'implicat primer $s^{*e}_{0,2}$ quedarà totalment cobert per $s^{*e}_{0,2,4,6}$ i per tant passarà a ser un implicat primer absolutament no essencial, $s^{*e}_{0,2} \rightarrow s^{*a}_{0,2}$. Això ho podem veure en la Figura 3.23

L'expressió mínima POS és doncs la AND d'aquests dos implicats primers $s^{*e}_{0,2,4,6} = c$ i $s^{*e}_{2,3} = a + \overline{b}$ donant lloc a,

$$f_{\text{POS-min}} = (a + \overline{b})\, c$$

que en aquest cas és l'expressió mínima de la funció que es pot implementar amb 10 transistors mentre que per la SOP-min en necessitaríem 14.

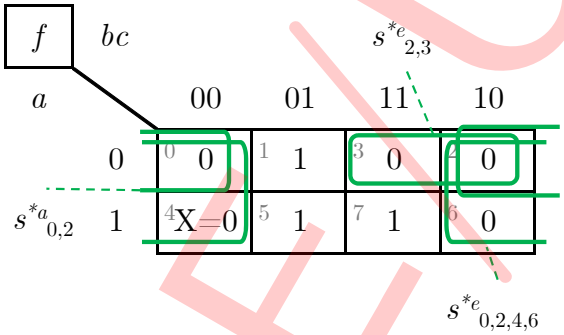


Figura 3.23: Implicants primers de la funció.

Capítol 4

Mètodes Automàtics de Minimització

En l'apartat anterior, en la presentació del mètode de minimització per Karnaugh, s'ha pogut veure que la limitació més gran és que el nombre de variables de la funció està limitat a quatre. Hi ha maneres d'estendre el mètode a cinc i fins i tot a sis variables però resulta ser molt poc eficient i farragós.

Com que la minimització de funcions és un pas clau en el procés de síntesi i implementació de circuits combinacionals, dels anys 50 als 80 es van desenvolupar mètodes eficients per la minimització de funcions combinacionals, fàcils d'automatitzar, i amb capacitat per tractar desenes de variables.

Els mètodes es poden separar en dos grans grups: els exactes i els aproximats. En els exactes la minimització és capaç de trobar l'expressió mínima absoluta que per tant té el mínim cost d'implementació en un circuit combinacional en dos nivells (SOP o POS). L'exemple més conegut és el **mètode tabular** o de **Quine-McCluskey**[1]. Aquest es va posteriorment incorporar en dos programes de CAD, el McBOOLE desenvolupat en la universitat de McGill [11], i l'ESPRESSO, fet al centre de recerca IBM Watson i la universitat de Berkeley [12]. Aquestes eines poden manipular funcions de fins a 20 variables. Amb un nombre major de variables el temps de minimització resulta poc pràctic.

En els mètodes aproximats [13], com per exemple el GREEDYCOV, s'accelera la cerca del mínim a base d'inspeccionar solucions mínimes locals. El resultat és que normalment no s'obté un mínim absolut de la funció però, en tot cas, un mínim que si aproximi força i que normalment serà vàlid com a alternativa de disseny. L'acceleració que s'obté amb els mètodes aproximats permet que el nombre de variables que pot

[1] Va ser inventat els anys 1950s per Willard V. Quine de la universitat de Harvard i J. McCluskey dels laboratoris AT&T Bell.

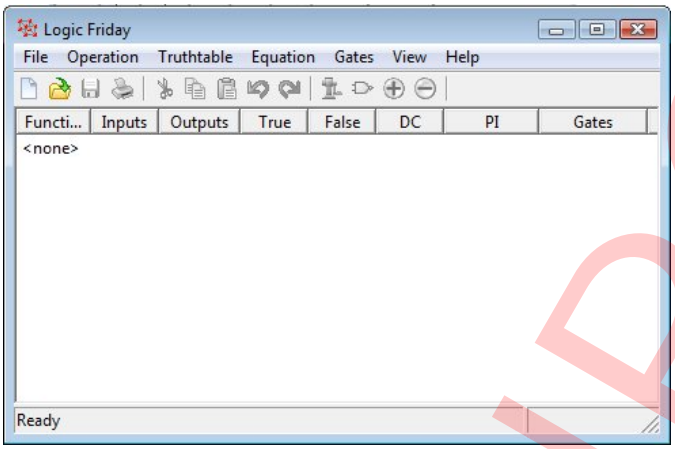


Figura 4.1: Interfície d'usuari del programa de síntesi lògica **Logic Friday**.

incloure la funció sigui superior a 20 i poden arribar a tractar amb un temps raonable algun centenar.

Les eines que CAD que incorporen aquests mètodes també inclouen altres millores en la minimització com són: minimització multi-funció i factorització, més endavant se'n veurà algun exemple.

4.1 Logic Friday

Logic Friday (LOFR) [14] és una eina de accés obert per síntesi lògica que incorpora ESPRESSO en el seu nucli a banda d'altres petites prestacions. Es pot executar en entorn windows o linux (emprant l'emulador wine). Incorpora alhora una interfície d'usuari amigable que facilita la definició de les funcions i l'obtenció i interpretació dels resultats.

En els propers apartats es farà una breu descripció de l'eina i s'il·lustraran alguns exemples amb els quals també es mostraran els beneficis de la minimització per grup i la factorització esmentades anteriorment.

En la Figura 4.1 es mostra la interfície d'usuari amb el LOFR. Els menus principals tenen les següents funcions:

- `File`: les opcions principals estan relacionades amb definir una nova funció, que es poden entrar en forma de TV, EL o esquema de portes. També hi ha opcions d'importació i exportació de TVs en format CSV, per poder-les tractar en un full de càlcul.
- `Operation`: permet minimitzar la funció, implementar-lo en portes lògiques i generar un programa C amb la TV incorporada. També hi ha opcions per

manipular diferents funcions prèviament definides com: comparar-les i fer lògica AND, OR, etc.

- `Truthtable`: disposa de totes les opcions per a la manipulació de les TV. Entrar una de nova, fixant 1ns/0s/Xs i validar la TV introduïda.
- `Equation`: similar a l'anterior però per les ELs. Es poden entrar expressions en format text i es poden seleccionar opcions de presentar les expressions resultants en forma SOP, POS o factoritzades.
- `Gates`: permet editar un diagrama de portes generat prèviament, fer-ne copia al porta papers, generar el llistat de components integrats, fer una traça dels valors lògics en el diagrama, a banda d'altres opcions de visualització.
- `View`: permet veure les formes simplificades o esteses de les TVs i ELs.
- `Help`: menú d'ajut.

Veiem-ne alguns exemples.

Exemple 25 En aquest exemple minimitzarem una funció de cinc variables i presentarem les diferents expressions lògiques que ens permet obtenir LOFR.

Accedim al menú (`File` → `New` → `Truth Table`). Ens demanarà que especifiquem el nombre de variables d'entrada (fixem 5) i la de funcions de sortida (fixem 1). També es poden editar els noms d'aquestes (prenem els que surten per defecte). Un cop acceptada aquesta pantalla inicial se'ns obre el patró de TV amb totes les files a "0". Amb el ratolí podem canviar el valor de les caselles fent doble o triple click a "1" o "X". A la Figura 4.2 es mostra la TV introduïda per aquest exemple.

Un cop premem la tecla de Retorn, LOFR avalua la TV i la presenta en forma compacta (mostrant les files diferents de "0") i també genera la EC-I. A través del menú (`Equation` → `Product of Sums`) podem obtenir també l'expressió EC-II, que s'il·lustra en la Figura 4.3.

Per tal de minimitzar les expressions, accedim al menú de (`Operation` → `Minimize...`). Se'ns presenta una finestra on es demana el mètode (aproximat o exacte), en aquest cas hem triat el segon. Un cop validat ja ens apareix l'expressió SOP-min i podem visualitzar també la POS-min, les teniu a la Figura 4.4.

Anem a veure un altre exemple de la minimització per grup. Suposem que tenim dues funcions que comparteixen les mateixes variables. Es pot demostrar que si minimitzem cadascuna de les dues funcions individualment, la implementació final pot resultar més costosa que si es fa una minimització per grup[2]. Això és degut a què en la minimització per grup s'afavoreixen els termes productes que poden ser compartits per les dues funcions.

[2]En aquest document no es tracta aquesta metodologia de minimització però es pot trobar en la bibliografia adjunta [2](“Multiple-Output Functions”, pg. 330).

Term	A	B	C	D	E	=>	F0
0	0	0	0	0	0		X
1	0	0	0	0	1		0
2	0	0	0	1	0		1
3	0	0	0	1	1		0
4	0	0	1	0	0		1
5	0	0	1	0	1		1
6	0	0	1	1	0		0
7	0	0	1	1	1		0
8	0	1	0	0	0		0
9	0	1	0	0	1		1
10	0	1	0	1	0		0
11	0	1	0	1	1		1
12	0	1	1	0	0		1
13	0	1	1	0	1		0
14	0	1	1	1	0		0
15	0	1	1	1	1		0
16	1	0	0	0	0		X
17	1	0	0	0	1		0
18	1	0	0	1	0		X
19	1	0	0	1	1		0
20	1	0	1	0	0		0
21	1	0	1	0	1		1
22	1	0	1	1	0		0
23	1	0	1	1	1		1
24	1	1	0	0	0		0
25	1	1	0	0	1		0
26	1	1	0	1	0		1
27	1	1	0	1	1		1
28	1	1	1	0	0		0
29	1	1	1	0	1		0
30	1	1	1	1	0		1
31	1	1	1	1	1		0

Figura 4.2: Taula de la veritat de la funció `F0(A,B,C,D,E)`.

A	B	C	D	E	=>	F0
0	0	0	0	0		X
0	0	0	1	0		1
0	0	1	0	0		1
0	0	1	0	1		1
0	1	0	0	1		1
0	1	0	1	1		1
0	1	1	0	0		1
1	0	0	0	0		X
1	0	0	1	0		X
1	0	1	0	1		1
1	0	1	1	1		1
1	1	0	1	0		1
1	1	0	1	1		1
1	1	1	1	0		1

```
Entered by truthtable:
F0 = A' B' C' D E' + A' B' C D' E' + A' B' C D' E +
A' B C' D' E + A' B C' D E + A' B C D' E' + A B' C
D' E + A B' C D E + A B C' D E' + A B C' D E + A B
C D E';

Unminimized Product of Sums:
F0 = (A+B+C+D+E')(A+B+C+D'+E')(A+B+C'+D'+E)
(A+B+C'+D'+E')(A+B'+C+D+E)(A+B'+C+D'+E)
(A+B'+C'+D+E')(A+B'+C'+D'+E)(A+B'+C'+D'+E')
(A'+B+C+D+E')(A'+B+C+D'+E')(A'+B+C'+D+E)
(A'+B+C'+D'+E)(A'+B'+C+D+E)(A'+B'+C+D+E')
(A'+B'+C'+D+E)(A'+B'+C'+D+E')(A'+B'+C'+D'+E');
```

Figura 4.3: Taula de la veritat en forma compacta i expressions EC-I i EC-II.

```
Minimized:
F0 = B' C' E' + A B' C E + A' B C' E + A B D E' +
A' C D' E' + B C' D E + B' C D' E;

Minimized Product of Sums:
F0 = (B'+C'+E')(A+C'+D')(A'+B'+D)(B+C+E')(A'+B+E)
(A+B'+C+E);
```

Figura 4.4: Expressions SOP-min i POS-min.

Term	A	B	C	D	=>	F0	F1	F2	F3
0	0	0	0	0		0	0	0	0
1	0	0	0	1		0	1	0	1
2	0	0	1	0		1	0	1	0
3	0	0	1	1		0	0	0	0
4	0	1	0	0		1	1	0	1
5	0	1	0	1		1	1	0	1
6	0	1	1	0		1	1	0	0
7	0	1	1	1		1	1	0	0
8	1	0	0	0		X	1	1	0
9	1	0	0	1		0	0	0	1
10	1	0	1	0		0	1	1	X
11	1	0	1	1		0	0	0	1
12	1	1	0	0		X	1	X	1
13	1	1	0	1		0	0	0	1
14	1	1	1	0		0	0	1	0
15	1	1	1	1		X	0	0	1

Figura 4.5: Taulta de la veritat de les quatre funcions.

```
Minimized:
F0 = A' B  + A' C D';
F1 = A' B  + A' C' D + A B' D' + B C' D';
F2 = A D' + B' C D';
F3 = B C'  + A D + C' D;
```

Figura 4.6: Expressions SOP-min individuals de cada funció.

Exemple 26 Disposem de quatre funcions {`F0,F1,F2,F3`} que comparteixen les quatre variables (`A,B,C,D`). La TV que especifica les quatre funcions la podem veure a la Figura 4.5.

Procedim a fer la minimització de les funcions. En la finestra de selecció del tipus de minimització cliquem a (`Mode` → `Exact`) i (`Multiple Outputs` → `Minimize each output independently`). Les expressions SOP-min que obtenim les podem veure a la Figura 4.6.

Si identifiquem els termes productes que hem de generar observem que només un (`A'B`) es repeteix en les funcions `F0` i `F1` i la resta són diferents. Això vol dir que en la implementació del circuit caldrà generar un total de 10 termes productes que, si incloem la inversió de les variables i les portes OR per les sortides, dóna un total de 80 transistors. En la Figura 4.7 es pot veure el circuit lògic.

Repetim el procés de minimització però ara seleccionant les opcions (`Mode` → `Exact`) i (`Multiple Outputs` → `Minimize jointly for smallest number of product terms`). Les equacions SOP-min les trobem a la Figura 4.8. Del total d'onze termes producte veiem que únicament n'hi ha set de diferents i quatre de repetits que són: (`A'B`), (`A'B'CD'`), (`A'C'D`) i (`BC'D'`). El nombre de transistors requerits per aquest implementació és de 68. En la Figura 4.9 es pot veure l'esquema de portes.

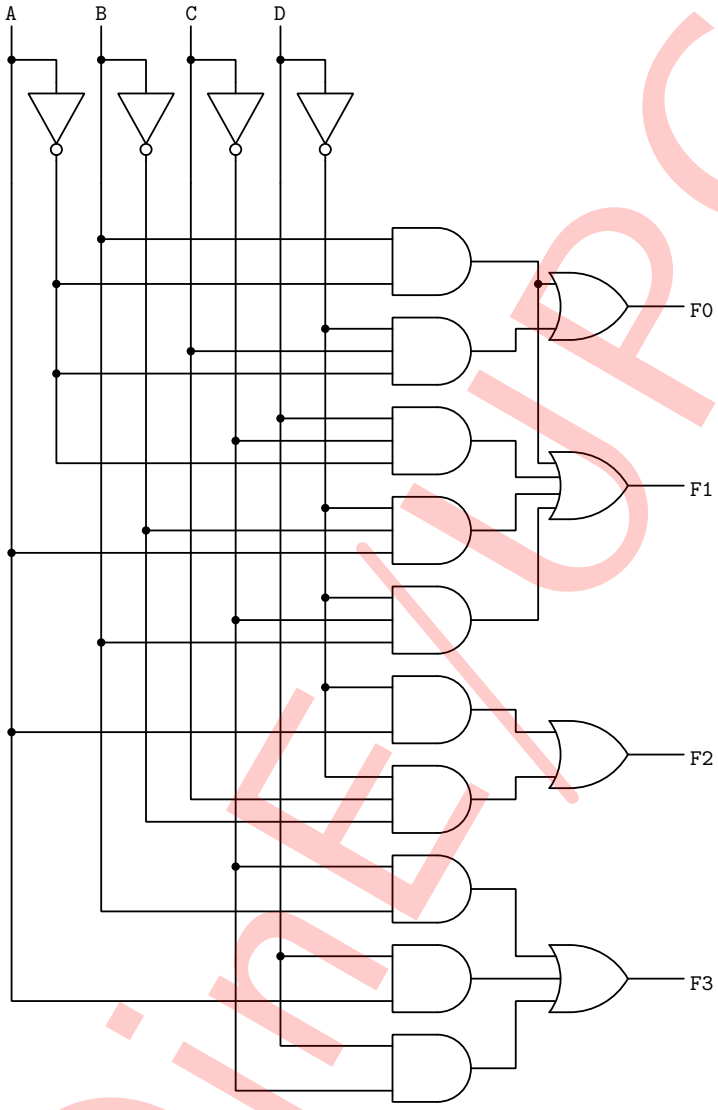


Figura 4.7: Implementació del circuit lògic fet a partir de les expressions minimitzades individualment. El nombre de transistors requerits és de 80.

```
Minimized:
F0 = A' B' C D' + A' B ;
F1 = A B' D' + A' C' D + B C' D' + A' B ;
F2 = A' B' C D' + A D';
F3 = A' C' D + A D + B C' D';
```

Figura 4.8: Expressions SOP-min minimitzades per grup.

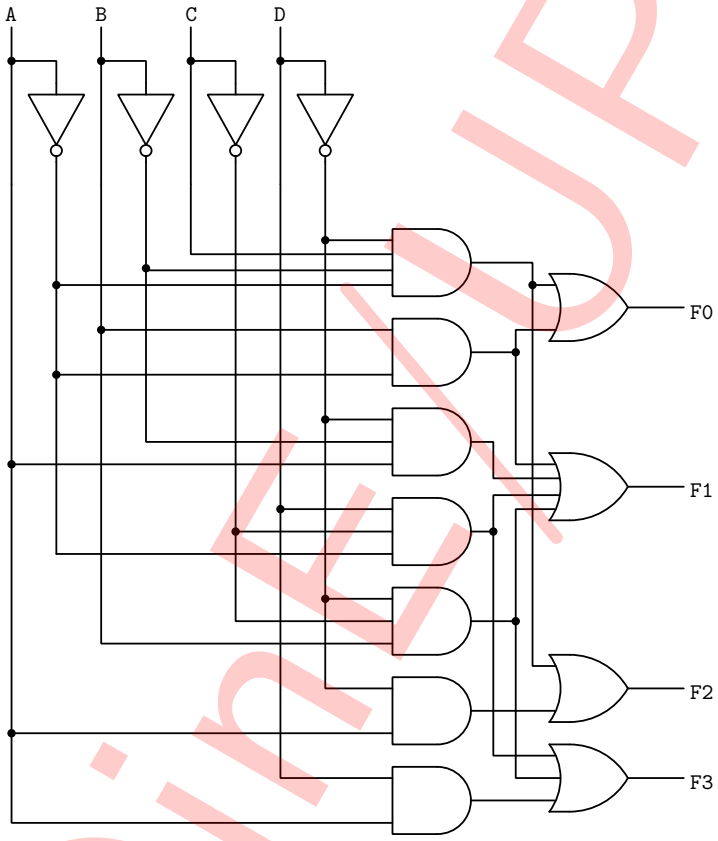


Figura 4.9: Implementació del circuit lògic fet a partir de les expressions minimitzades en grup. El nombre de transistors requerits és de 68.

```
Entered by truthtable:
F0 = A' B' C' D' + A' B' C' D + A' B' C D' + A' B' C D +
A' B C' D' + A' B C D' + A B' C' D + A B' C D' + A B C' D
+ A B C D' + A B C D;
```

Figura 4.10: Expressió canònica SOP de la funció.

```
Minimized:
F0 = A' B'  + A' D' + C D' + A B D + A C' D;
```

Figura 4.11: Expressió SOP-min de la funció.

Veiem un darrer exemple. En aquest emprarem l'opció de factorització. Aquesta transformació extreu literals comuns dels termes productes de l'expressió SOP i els agrupa emprant la propietat distributiva de manera que augmenta el nombre de nivells de l'expressió i en conseqüència també el nombre de nivells del circuit. Té l'avantatge de què es redueix el nombre de transistors de la implementació però l'inconvenient que el temps de resposta del circuit serà una mica més lent degut a que els senyals hauran de travessar més nivells de portes.

Exemple 27 Suposem que tenim la funció lògica de quatres variables `F0(A,B,C,D)` especificada per l'expressió canònica EC-I mostrada a la Figura 4.10.

Seguint el mateix procediment que en els casos anteriors minimitzem l'expressió i obtenim la SOP-min que es pot veure en la Figura 4.11. Si comptabilitzem el nombre de transistors necessaris per a implementar el circuit obtindrem un total de 42.

Ara seleccionem en el menú del LOFR (`Equation` → `Factor`). La nova expressió que obtenim es mostra a la Figura 4.12. S'han extret com a factor comú els

```
Factored:
F0 = A D (C' + B) + D' (C + A') + A' B';
```

Figura 4.12: Expressió SOP-min de la funció.

literals `A`, `D` i `D'` transformant l'expressió en una de tres nivells de portes. La implementació d'aquesta requereix de 36 transistors, 6 menys que en l'expressió SOP-min. El circuit en portes lògiques es pot veure en la Figura 4.13 i el nombre de transistors es pot comptabilitzar a partir de comptar les entrades a les portes multiplicant-ho[3] per 2. Aquest estalvi extra de transistors en relació a l'expressió mínima però, té un cost en temps de propagació del senyal, ja que en la versió del circuit factoritzada els senyals hauran de travessar 1 nivell més de portes i per tant trigaran un 33% més del temps aproximadament.

[3] Assumint una tecnologia CMOS.

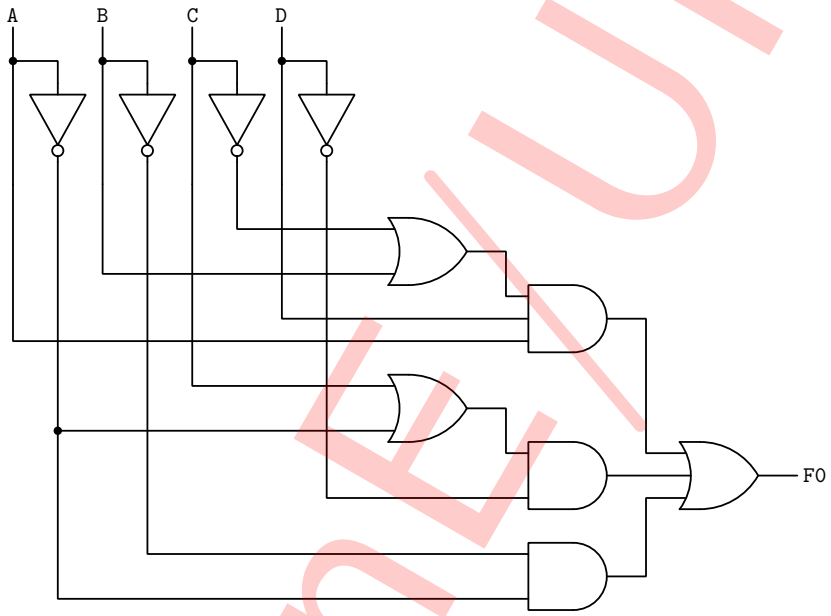


Figura 4.13: Implementació del circuit lògic fet a partir de l'expressió factoritzada.

Capítol 5

Conclusions

En aquest quadern s'ha presentat una introducció als sistemes combinacionals que tenen la peculiaritat de no dependre del temps. La seva resposta és únicament funció de les variables binàries d'entrada i el seu temps de resposta instantani, en condicions ideals. Aquests, estan descrits per funcions lògiques o combinacionals que pertanyen a l'àlgebra de commutació.

En el segon capítol, s'ha explicat l'especificació de les funcions lògiques, amb les **Taules de la Veritat** i amb les **Expressions Lògiques**. S'han descrit les expressions canòniques suma-de-productes i producte-de-sumes i la manera de generar-les a partir de la Taula de la Veritat. S'han fet exemples de com es fa una implementació amb portes lògiques a dos nivells a partir d'una expressió lògica suma-de-productes o producte-de-sumes.

En el tercer capítol, s'ha introduït la metodologia de minimització de les expressions lògiques. Aquesta persegueix com a objectiu l'obtenció d'una expressió lògica que doni lloc a una implementació a dos nivells amb el nombre mínim de transistors. La metodologia de minimització s'ha basat en **mètode de Karnaugh** o **mapes-K**. S'ha presentat l'estructura del mapa-K i s'han identificat les agrupacions de 1ns i 0s amb els termes productes i sumes de les expressions. S'han especificat els implicants i implicats primers de les expressions i com identificar-los en el mapa-K. S'ha explicat la metodologia de com seleccionar aquests termes primers a fi de generar les expressions mínimes suma-de-productes i producte-de-sumes. S'ha il·lustrat la metodologia amb diversos exemples.

En el darrer capítol s'han descrit els mètodes automàtics de minimització, els **exactes** i els **aproximats**. S'ha presentat una eina de codi obert que implementa ambdós tipus de mètodes (Logic Friday). S'han vist diversos exemples de minimització amb l'eina, en concret amb més de quatre variables i per diverses funcions de sortida. S'ha vist la minimització en grup i els beneficis de la seva optimització en front de la minimització individual de funcions. S'ha il·lustrat també els beneficis de la

factorització de les expressions pel que fa a la disminució del nombre de transistors requerits en la implementació amb la relació cost-benefici pel que fa al temps de resposta del circuit.

Bibliografia

[1] R. M. Gray, *Entropy and information theory.* Springer Science & Business Media, 2011.

[2] J. P. Hayes, *Introduction to digital logic design.* Addison-Wesley Longman Publishing Co., Inc., 1993.

[3] V. P. Nelson, H. T. Nagle, B. D. Carroll, and D. J. Irwin, *Análisis y diseño de circuitos lógicos digitales.* Prentice Hall, 1996.

[4] J. F. Wakerly, *Digital design.* fourth edition, Prentice Hall, 2007.

[5] G. Boole, *The laws of thought.* Open Court Publishing Company, 1916, vol. 2.

[6] C. E. Shannon, "A symbolic analysis of relay and switching circuits," *Electrical Engineering*, vol. 57, no. 12, pp. 713–723, 1938.

[7] E. V. Huntington, "New sets of independent postulates for the algebra of logic, with special reference to whitehead and russell's principia mathematica," *Transactions of the American Mathematical Society*, vol. 35, no. 1, pp. 274–304, 1933.

[8] D. Gries and F. B. Schneider, *A logical approach to discrete math.* Springer Science & Business Media, 2013.

[9] N. H. Weste and K. Eshraghian, *Principles of CMOS VLSI design.* Addison-Wesley New York, 1985, vol. 188.

[10] M. Karnaugh, "The map method for synthesis of combinational logic circuits," *Transactions of the American Institute of Electrical Engineers, Part I: Communication and Electronics*, vol. 72, no. 5, pp. 593–599, 1953.

[11] M. R. Dagenais, V. K. Agarwal, and N. C. Rumin, "Mcboole: A new procedure for exact logic minimization," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 5, no. 1, pp. 229–238, 1986.

[12] R. K. Brayton, G. D. Hachtel, C. McMullen, and A. Sangiovanni-Vincentelli, *Logic minimization algorithms for VLSI synthesis*. Springer Science & Business Media, 1984, vol. 2.

[13] V. Chvatal, "A greedy heuristic for the set-covering problem," *Mathematics of operations research*, vol. 4, no. 3, pp. 233–235, 1979.

[14] S. Rickmann, "Logic friday [computer software]," https://sontrak.com/, 2011.