Universitat Autònoma de Barcelona

Mapatge Tecnològic Orientat a Generadors de Mòduls

Memòria presentada per
Jordi Riera i Baburés
per optar al grau de
Doctor en Informàtica

Bellaterra, Juliol de 1996

Jordi Carrabina i Bordoll, profesor Titular de Universitat de l'àrea de Arquitectura i Tecnologia de Computadors del Departament d'Informàtica d'aquesta Universitat,

CERTIFICA :

que la present memòria ha estat realitzada sota la seva direcció per en Jordi Riera i Baburés, constituint la tesi doctoral per accedir al Grau de Doctor en Informàtica.

Bellaterra, Juliol de 1996

Jordi Carrabina i Bordoll

Agraïments

Començo per agrair la col·laboració, amistat i suport dels membres de la Unitat de Microelectrònica de la Université Catholique de Louvain-la-Neuve, en especial d'en Ricardo Jacobi i la Doctora Anne-Marie Trullemans, per iniciar-me en l'intricat món de la síntesi lògica.

A l'André Reis per les fructíferes discussions sobre la implementació de funcions Booleanes usant BDDs ordenats o no ordenats.

Voldria esmentar i agrair especialment el suport dels meus companys i amics de la Unitat de Microelectrònica, sense els quals a ben segur aquest treball no seria el que és. En especial a en Lluís Ribas i a en Josep Velasco pels comentaris i l'ajuda rebuda, sobretot en els moments més durs, on el futur semblava un abisme sense fi.

També voldria fer esment de la col·laboració incondicional del meu director, el Doctor Jordi Carrabina, que ha fet quelcom més que les tasques de direcció. La seva empenta ha estat decisiva per tirar endavant aquest treball.

Al Doctor Jordi Aguiló i, per extensió, al Departament de Disseny i al Centre de Càlcul del Centre Nacional de Microelectrònica de Barcelona, per les facilitats presentades en el transcurs del desenvolupament d'aquest treball.

No voldria oblidar tampoc tota aquella gent que han participat d'alguna manera o altra en l'elaboració i desenvolupament d'aquest treball.

A la meva estimada Imma

... per les valuoses hores que aquesta tesi li ha robat.

Taula de Continguts

Capítol 1

Introducció

Les especificacions dels circuits integrats actuals sovint són tant restrictives que semblen difícilment assolibles sense una automatització creixent del procés de disseny. Així, l'automatització del procés de disseny provocà un canvi en la metodologia de disseny original, en la qual tots els components d'un circuit integrat es dissenyaven a mà fins al nivell de descripció geomètrica. S'introduïren les metodologies de disseny basades en biblioteques de cel·les estàndard predissenyades i precaracteritzades. Al mateix temps, s'introduïren eines automàtiques de posicionament i connexionat d'aquestes cel·les, de tal manera que el dissenyador podia treballar en un nivell més alt d'abstracció, oblidant-se dels penosos detalls del nivell físic, massa proper de la tecnologia. El desig d'augmentar el nivell d'abstracció en el qual ha de treballar el dissenyador, va provocar ben aviat l'aparició d'eines de síntesi, lògica i física, i verificació automàtica de circuits integrats. Aquestes eines portaven a un major grau d'abstracció dels detalls tecnològics i permetien al dissenyador concentrar-se en els nivells superiors del procés de disseny. Les eines de síntesi s'encarregaven de traduir les descripcions d'alt nivell proporcionades pel dissenyador a un circuit implementat en base a les cel·les disponibles en una determinada biblioteca. Aquest procés d'automatització ha anat evolucionant fins als nostres dies, on la síntesi automàtica es pot fer des d'alts nivells d'abstracció: a nivell de comportament o a nivell de sistema *hardware/software*.

L'evolució de les eines de síntesi automàtica ha anat acompanyada d'una evolució paral·lela de la tecnologia de fabricació de circuits integrats que ens ha portat fins a les actuals tecnologies submicròniques o profundament submicròniques. La reducció de les mides dels dispositius tecnològics augmenta la densitat d'integració. Més i més dispositius es poden encabir en un mateix circuit integrat, amb el consegüent augment de la complexitat del procés de disseny d'aquest circuits. A més a més, aquest augment de la densitat d'integració ha fet que problemes que abans afectaven a un conjunt molt petit de dissenys passin a ser ara problemes generals. Segurament l'exemple més clar és la dissipació de potència, que ha esdevingut un dels paràmetres crítics en el procés de disseny, al costat, o fins i tot per davant, del retard màxim del circuit o l'àrea ocupada.

Les especificacions més restrictives exigides als circuits actuals han fet que les funcions de cost tradicionalment utilitzades en les eines de síntesi automàtica no

fossin prou acurades per permetre assolir els requeriments exigits. Això obliga a costosos processos iteratius, on el cost del circuit s'extreu a partir de la informació disponible en el nivell físic i es realimenta als processos de síntesi. Aquesta realimentació pot no convergir si les funcions de cost usades en les eines de síntesi no modelen prou bé els objectius de disseny. O, simplement, el conjunt limitat de cel·les disponibles en una biblioteca pot fer inviable el circuit amb les especificacions donades. La solució proposada per a aquest problema ha estat dotar les biblioteques de cel·les de diferents versions de cada cel·la, cadascuna d'elles amb diferents objectius de disseny (àrea, retard, dissipació de potència). No obstant això, el disseny i manteniment d'una biblioteca de cel·les estàndard és un procés costós que ha de ser repetit cada vegada que algun paràmetre de la tecnologia canvia.

El present treball s'emmarca dins d'una metodologia totalment diferent. Considerem un procés de síntesi lògica lligat a un generador de mòduls guiat per prestacions. És evident que el generador de mòduls disposarà d'informació tecnològica molt detallada i, per tant, serà capaç d'assolir compromisos molt més restrictius sobre els objectius de disseny. En concret, l'etapa de dimensionament de transistors és capaç de generar qualsevol versió implementable d'una porta. Aquest fet contrasta amb el número limitat de versions d'una porta que poden estar disponibles en una biblioteca de cel·les estàndard. D'altra banda, caldrà disposar d'una eina de síntesi lògica capaç de generar qualsevol porta implementable en una tecnologia determinada. D'aquesta manera la flexibilitat i suavitat de moviments en l'espai de solucions augmenta considerablement, cosa que garanteix l'acompliment de restriccions de disseny molt més estrictes que les que són assolibles en una metodologia basada en biblioteques de cel·les estàndard.

La resta del treball està organitzat de la següent manera :

En el capítol 2 es defineixen els conceptes matemàtics necessaris per a la total comprensió del present treball. S'introdueixen els conceptes relacionats amb l'Àlgebra de Boole, els Diagrames de Decisió Binària i les Xarxes Booleanes.

En el capítol 3 es presenta l'evolució de la metodologia de disseny al llarg de la història. Es mostra la creixent automatització del procés de disseny i la introducció de la síntesi automàtica a diferents nivells del procés de disseny dels circuits integrats. En el camp de la síntesi lògica, es presenta un estudi comparatiu de l'ús de biblioteques de cel·les bàsiques o de generadors de mòduls basats en transistors.

En el capítol 4 s'estudia el problema de generar una porta complexa, d'una sola etapa, que implementi una funció Booleana qualsevol representada per un BDD. S'estudia la possibilitat d'obtenir la implementació directament a partir de l'estructura del BDD. Es defineix el conjunt de funcions per a les quals aquesta implementació directa és possible. També es defineix el conjunt de funcions per a les quals la implementació directa a partir de l'estructura del BDD resulta ser la implementació mínima. La resta de funcions seran implementades a partir d'una factorització màxima de la funció.

En el capítol 5 es tracta el procés d'agrupament dels nodes de la Xarxa Booleana. Per cada node de la Xarxa Booleana es pretén crear una llista amb tots els possibles agrupaments de funcions que es troben en el seu ventall transitiu d'entrades i són

implementables. En aquest capítol s'estudia la complexitat d'aquest procés i es proposen heurístiques per tal de reduir-la.

En el capítol 6 s'estudia el procés de cobertura d'una Xarxa Booleana bo i minimitzant una funció de cost que implica un compromís, proposat per l'usuari, entre els diferents objectius de disseny. S'analitzen diverses estratègies de cobertura de la Xarxa Booleana i es comparen els resultats d'aplicar-ne unes o altres.

Finalment, en el capítol de conclusions s'analitzen les contribucions del treball presentat en aquesta tesi en el camp de la síntesi lògica orientada a generadors de mòduls guiats per prestacions i s'indiquen també les línies de futur considerades més interessants.

Capítol 2

Definicions i Notació

En aquest capítol es defineixen els conceptes matemàtics necessaris per a la total comprensió del present treball. S'introdueixen els conceptes relacionats amb l'Àlgebra de Boole, els Diagrames de Decisió Binària i les Xarxes Booleanes.

Introducció

Les definicions i les notacions relacionades amb els temes que es descriuen en aquest capítol parteixen de l'assumpció d'unes idees matemàtiques bàsiques i d'un lèxic relacionat que s'ofereix tot seguit.

Aquest apartat és una adaptació del capítol primer de [**Bro90**]. L'estructuració de la resta d'apartats segueix una certa evolució del que s'indica aquí: des de l'àlgebra de Boole fins les Xarxes Booleanes, passant pels diagrames de decisió binària com a representacions efectives de funcions de commutació, que es relacionen de manera directa amb les funcions Booleanes.

Una *fórmula* o *expressió* és una seqüència de símbols d'un llenguatge concret, sobre els que es defineix una sintaxi i una semàntica. La sintaxi estableix l'ordenació dels símbols en l'expressió i la semàntica en defineix el seu significat.

Una *proposició* és una fórmula que és necessàriament certa o falsa, mentre que un *predicat* és una fórmula que, per determinades substitucions de símbols, pot esdevenir una proposició (és a dir, ser sempre certa o falsa). Per això, a vegades un predicat *P* s'anomena també una *forma proposicional*. Sigui *P* un predicat:

$$P(X)\ ; \quad \text{a on} \quad X = x_0,\ldots,x_{n-1}$$

El domini *D* del predicat P és el conjunt de valors que es pot assignar a *X*. Un *quantificador* (∀, ∃) restringeix el domini *D* a on *P* ha de ser cert. El quantificador universal, ∀, indica que *P* ha se ser cert per tot element del domini, mentre que el

quantificador existencial, ∃, indica que P ha ser cert com a mínim per un element del domini.

S'estableixen dues relacions entre els predicats, la d'*implicació* i la d'*equivalència*: $P(X) \Rightarrow Q(X)$ i $P(X) \Leftrightarrow Q(X)$ respectivament. Si P implica Q llavors no pot passar que P sigui cert i Q fals per a cap X de D. En l'altre cas, que P sigui equivalent a Q suposa que P i Q són certs o falsos per als mateixos X de D.

Abans de relacionar les fórmules i les funcions, repassarem breument els *conjunts*, que són col·leccions d'objectes, els quals s'anomenen *elements* del conjunt. Aquests conjunts es poden descriure per enumeració com, per exemple, a la següent fórmula: {roig, verd, blau}. També es pot determinar una propietat de pertanyença: $S=\{x|P(x)\}$, a on $P(x)$ és cert si $x \in S$. Una tercera forma de descripció és a través d'una regla de recursió, en la qual es distingeixen dos components: la base i la recursió. En la base s'enumeren els elements primigenis del conjunt i la recursió indica la forma de construir altres elements que també formen part del conjunt.

Cal recordar que els conjunts estan formats per elements diferents i no ordenats, a diferència de les *seqüències*, en les quals l'ordre d'enumeració dels elements és important i en les quals els elements es poden repetir. A vegades, es parla de conjunts ordenats, n-tuples, taules o vectors.

Dos conjunts són *iguals* si tots dos contenen els mateixos elements. Un conjunt S està *inclòs* en un altre conjunt T si tots els elements de S són també de T:

$$[S \subseteq T] \Leftrightarrow (\forall x)[x \in S \Rightarrow x \in T]$$

La *cardinalitat* es defineix com el número d'elements en un conjunt finit, denotat per $|S|$. El *conjunt buit*, ∅, és aquell que té cardinalitat zero i, òbviament, és inclòs a qualsevol altre conjunt.

Les operacions bàsiques que es poden fer amb conjunts són el producte cartesià, la unió, la intersecció i el complement. El complement de T relatiu a S és $S–T$ i el complement absolut de T, $\overline{T}$, és $U–S$, a on U és el *conjunt universal*; és a dir, aquell que conté tots els conjunts d'interès. Les *particions*, $\pi_S = \{B_1, ..., B_k\}$, són conjunts de subconjunts no buits de S, la intersecció dels quals és nul·la i la unió dels quals dóna S.

Les *relacions* R entre dos conjunts S i T es poden veure com a subconjunts de productes cartesians:

$$xRy \Leftrightarrow (x,y) \in R, \text{ a on } x \in S \text{ i } y \in T$$

Una relació es pot definir entre elements d'un mateix conjunt. En aquest cas pot ser una relació d'*equivalència* si R acompleix les propietats reflexiva, simètrica i transitiva; o d'*ordre parcial* si acompleix l'antisimètrica enlloc de la simètrica.

Una *funció* f d'un conjunt S cap a un conjunt T, $f:S \to T$, assigna a cada element $x \in S$ un element $f(x) \in T$ anomenat imatge de x. El conjunt S s'anomena *domini* de la

funció i el *T* el codomini de *f*. El *rang* de *f* és el conjunt d'imatges de *S* sota *f* i és, clarament, un subconjunt del codomini.

Una funció és una relació especialitzada, és a dir, la podem veure com una relació de *S* a *T* en la que cada element de *S* apareix només un cop com a primer element d'un dels parells ordenats de *f* o $S \times T$. Així, les fórmules $f(x) = y$ i $(x, y) \in f$ són predicats equivalents. Existeixen *funcions de n variables*, $f : S^n \to T$ en les quals els dominis són productes cartesians d'ordre *n* de *S*. I també, funcions proposicionals $f : S \to T$ en les quals el codomini és $\{cert, fals\}$.

Una fórmula com $f(x) = x^2 - 9$ resulta més senzilla d'utilitzar que no un conjunt de parells ordenats que descrigui la funció *f*. De totes maneres, cal remarcar que són dues entitats diferents i que les fórmules són, en tot cas, representacions de funcions. Per tant, *normalment existiran diverses fórmules que representin una mateixa funció*. A nivell de sistemes digitals, es pot associar el comportament a una funció Booleana i l'estructura a una fórmula Booleana.

Per acabar amb aquesta introducció, es repassarà breument què és una operació i un sistema algebraic: una *operació*, *op*, sobre un conjunt *S* és una funció $f : S \times S \to S$. El parell (*S*, *op*) és un *sistema algebraic*. Una *àlgebra de Boole* és un sistema algebraic de la forma (*B*, +, ., 0, 1), a on *B* és el conjunt, +, - són les operacions, i els últims dos elements són membres especials de *B*. Cal remarcar que existeixen altres formes particulars d'omplir aquesta quíntupla que també formen àlgebres de Boole. Per exemple, el sistema algebraic següent: (2^S, $\cup$, $\cap$, $\varnothing$, *S*) és una àlgebra de Boole, on 2^S consisteix en el conjunt dels subconjunts de *S*.

Àlgebra de Boole

Aquest tema ofereix de manera breu tots els atributs que té una àlgebra de Boole, així com les seves propietats. La informació s'ha recollit del capítol preliminar de [**LB94**].

Una àlgebra de Boole és una estructura matemàtica (*B*, +, ·, 0, 1) que satisfà les següents propietats: si $x, y, z \in B$, llavors

$$x + x = x \qquad x \cdot x = x$$

$$x + (y + z) = (x + y) + z \qquad x \cdot (y \cdot z) = (x \cdot y) \cdot z$$

$$x + y = y + x \qquad x \cdot y = y \cdot x$$

$$x + (x \cdot y) = x \qquad x \cdot (x + y) = x$$

$$x \cdot (y + z) = x \cdot y + x \cdot z \qquad x + (y \cdot z) = (x + y) \cdot (x + z)$$

Existeixen dos elements distingits, la identitat, *1*, i el nul, *0*, tal que:

$$x + 1 = 1,\ x \cdot 0 = 0, \quad x \cdot 1 = x,\ x + 0 = x$$

Per a cada *x*, existeix un $\bar{x}$ tal que:

$$x+\bar{x}=1, \quad x\cdot\bar{x}=0$$

Una *funció Booleana* és una $f(X)\colon B^n \to B$, a on $B=\{0,1\}$ i $X=(x_1,\ldots,x_n)$. Cada $x_i \in X$ és una variable Booleana que pot prendre els valors *0* o *1*.

Una *funció lògica* és una $f(X)\colon B^n \to \{B \cup *\}$, a on s'inclou un nou element, *, en el codomini per indicar els casos on no importa el valor que pren la funció. Aquestes funcions lògiques es caracteritzen per dues funcions Booleanes:

$$g\colon B^n \to B \text{ tal que } f(X)=1 \Leftrightarrow g(X)=1$$

$$h\colon B^n \to B \text{ tal que } f(X)=0 \Leftrightarrow h(X)=0$$

Respecte de les funcions lògiques *f*, es defineixen els següents termes:

- El *conjunt actiu* de $f(X)$ és $g^{-1}(1)$ i el *conjunt passiu* és $h^{-1}(0)$. Normalment, se'ls anomena *on-set* i *off-set*, respectivament. Pot existir també un conjunt d'*indiferents* o *don't-care-set* definit per $B^n-\left(g^{-1}(1)+h^{-1}(0)\right)$.

- $f(X)$ és *completament especificada* si el conjunt d'indiferents és buit. D'altra manera, és *incompletament especificada*. En aquest treball es consideraran només funcions completament especificades i, per tant, les referències a les funcions lògiques són equivalents a les referències a funcions Booleanes (el conjunt imatge és {0,1}).

- El *complement* de $f(X)$, $\overline{f(X)}$, s'obté intercanviant el conjunt actiu pel conjunt passiu.

- $f(X)$ és una *tautologia* si $f(X)\equiv 1$.

- El *cofactor* de $f(X)$ respecte de x_i, f_{x_i}, s'obté de $f(X)$ fent $x_i=1$. Similarment, $f_{\overline{x_i}}$ es treu fent $x_i=0$.

- El *suport* de $f(X)$, $\sup(f)$, és el conjunt de variables a $f(X)$, i el *mínim suport* d'$f(X)$, $\min\_\sup(f)$, és $\{x_i \mid f_{x_i} \neq f_{\overline{x_i}}\}$. En el present treball, si no s'especifica altrament, es parlarà del suport d'una funció referint-se al mínim suport de la funció.

- La *composició de funcions*, notada $f_1\big|_{x_i=f_2}$ és la funció que s'obté substituint tots els x_i a f_1 per f_2.

- La *descomposició* d'una funció Booleana *f(X)* consisteix en reexpressar la funció en base a un conjunt de funcions $g_1, g_2, \ldots, g_m$ i, possiblement, algunes de les variables en el suport original: $f(X', g_1, g_2, \ldots g_m)$, on $X' \subseteq X$.

- L'operació de *col·lapsar* una funció *g(X')* dins una funció *f(X,g(X'))* consisteix en reexpressar la funció *f* sense *g*, mitjançant el fet d'expandir *g* en *f*, resultant en *f(Y),* on $Y \subseteq \{X \cup X'\}$. El suport de la funció resultant, $Y \subseteq \{X \cup X'\}$, posa de relleu el fet que els suports *X* i *X'* no tenen perquè ser disjunts i que la funció resultant del col·lapse de *g(X')* dins de *f(X)* no té perquè dependre de la unió de *X* i *X'*, ja que algunes variables del suport de les funcions originals poden no aparèixer en el suport mínim després del col·lapse.

- $f(X)$ és *monòtona creixent* per x_i si $f_{\overline{x}_i} = 1 \Rightarrow f_{x_i} = 1$; direm també que x_i és una variable de fase positiva de $f(X)$. Similarment, $f(X)$ és *monòtona decreixent* per x_i si $f_{x_i} = 0 \Rightarrow f_{\overline{x}_i} = 0$; direm que x_i és una variable de fase negativa de $f(X)$.

- $f(X)$ és *monofàsica* en x_i, o de fase no binària, si $f(X)$ és monòtona creixent o decreixent en x_i. En cas contrari, $f(X)$ és *bifàsica* en x_i, o de fase binària en x_i. Direm que $f(X)$ és de *fase positiva (fase negativa)* si és monòtona creixent (decreixent).

També poden resultar útils els següents teoremes i propietats:

Teorema de l'*expansió de Boole-Shannon* [**Boo1847**, **Sha49**]: qualsevol funció Booleana $f(X)$ es pot expandir en alguna de les següents formes, per $x \in X$:

$$f(X) = x \cdot f_x + \overline{x} \cdot f_{\overline{x}} \tag{2-1}$$

$$f(X) = (\overline{x} + f_x) \cdot (x + f_{\overline{x}})$$

$$f(X) = x \cdot f_x \oplus \overline{x} \cdot f_{\overline{x}}$$

L'expansió més utilitzada és la definida per l'equació (2-1).

Pel que fa als cofactors, aquests compleixen les següents propietats. Siguin *f(X)* i *g(X)* funcions Booleanes i *x, y* elements de *X*.

$$(f_x)_y = (f_y)_x \qquad (f \cdot g)_x = f_x \cdot g_x$$

$$(\bar{f})_x = \overline{(f_x)} \qquad f \equiv 1 \Leftrightarrow f_x \equiv 1 \wedge f_{\overline{x}} \equiv 1$$

Cal fer notar que, si *f* és independent de *x*, llavors $f_x = f_{\overline{x}}$. En aquest cas, *x* no pertany al suport mínim de *f(X)*.

Cubs, cobertes i cobertures

Una variable Booleana pot agafar només dos valors, *0* o *1*. Un *literal* és una variable x o el seu complement $\bar{x}$. Un *cub* és una funció Booleana $\prod_{i\in\sigma} y_i$, a on $y_i \in \{x_i, \overline{x_i}\}$, i $\sigma \subseteq \{1,\ldots,n\}$. Un cub és un *minterm* quan $\sigma = \{1,\ldots,n\}$. Dualment, un *maxterm* és la funció Booleana $\sum_{i\in\sigma} y_i$, a on $y_i \in \{x_i, \overline{x_i}\}$.

Un cub *c* és cobert per un altre cub *d* si el conjunt actiu de *c* és contingut dins del conjunt actiu de *d*.

Un *implicant* d'una funció lògica *f* és un cub *c* el conjunt actiu del qual és inclòs en la unió del conjunt actiu i del conjunt d'indiferents de *f*. Un *implicant primer* és un cub *c* no cobert per cap altre implicant de *f*. I un *implicant primer essencial* és un cub que conté un minterm no inclòs en cap altre implicant de la funció.

Una *coberta*, *K*, és una funció Booleana $\sum_i c_i$, a on c_i és un cub. La representació d'una funció Booleana per una suma de cubs s'anonema *suma de productes (SOP)* o *representació a dos nivells*, ja que es correspon amb una realització d'un circuit amb dos nivells (AND-OR). Al contrari, una *representació multinivell*, o *forma factoritzada*, és una expressió algebraica amb parèntesis que es correspon a una Xarxa Booleana multinivell.

En relació a les cobertes, cal establir una certa nomenclatura, definicions i propietats que es descriuen en els següents paràgrafs.

- Una *cobertura (del conjunt actiu)* d'una funció lògica *f* la proporciona una coberta *K* que cobreixi tot el conjunt actiu de la funció i, possiblement, part del conjunt d'indiferents. Una descripció similar es pot aplicar a les *cobertures del conjunt passiu*.

- Una *coberta* o *cobertura primera* de *f* és la que només conté cubs o implicants primers de *f*.

- Una *coberta (cobertura) no redundant* és aquella de la qual no se'n pot retirar cap cub sense que deixi de ser una coberta (cobertura) de *f*.

- Les *cobertes monofàsiques* són les que només contenen cubs que són funcions monofàsiques.

- Una coberta és *mínima respecte la contenció de cubs únics* si cap cub és contingut dins d'algun altre.

- Una coberta *K* cobreix un cub *c* si i només si K_c és una tautologia.

Les funcions, cobertes i cobertures monofàsiques tenen especial importància en els processos de síntesi lògica multinivell, pel que és important comentar-ne algunes de les seves propietats:

- El complement d'una funció monofàsica és monofàsica. Si la funció és monofàsica de fase positiva, el seu complement serà de fase negativa, i a l'inrevés.
- Els cofactors d'una funció monofàsica són també monofàsics en la mateixa fase que la funció original.
- Cada primer implicant d'una funció monofàsica és essencial.
- Una cobertura primera d'una funció monofàsica és monofàsica en la mateixa fase que la funció original.
- Una coberta monofàsica és una tautologia si i només si conté un cub que és una tautologia.
- Una coberta monofàsica mínima (respecte de la contenció de cubs únics) és l'única coberta mínima.

BDDs

Existeixen moltes formes de representar una funció Booleana: suma de productes, suma de mínterms, producte de maxterms, etc. Direm que una representació és *canònica* si dues funcions idèntiques tenen la mateixa representació. Aquesta propietat és essencial per comprovar tautologies i per realitzar moltes altres operacions lògiques. A més, una representació hauria de ser *compacta i manipulable*. Totes aquestes característiques les acompleixen els *diagrames de decisió binària* (BDDs), sota una ordenació de variables fixada, que assegura la canonicitat.

La teoria relativa als BDDs es pot consultar en l'article d'en Bryant [**Bry86**], en el qual, a partir de les treballs preliminars de Lee [**Lee59**] i Akers [**Ake78**], desenvolupa aquesta nova representació. La resta de l'apartat resumeix el contingut de [**Bry86**] i les extensions proposades en [**BRB90**].

Un *BDD* és un graf acíclic dirigit (DAG) que representa una funció Booleana. Un BDD té una arrel i dues fulles o nodes terminals. L'arrel correspon a la funció representada pel BDD, i els nodes terminals corresponen a les funcions *0* o *1*. Cada node no terminal v té una variable associada x_i i dos nodes successors, v^V i v^F, anomenats fills. La funció representada per un node no terminal v d'un BDD, f^v, és $f^v = x_i \cdot f^{v^V} + \overline{x_i} \cdot f^{v^F}$ on x_i és la variable associada al node. Totes les variables han de respectar una ordenació parcial entre elles de tal manera que la variable associada a un node ha d'ocupar una posició en l'ordenació més gran que la que ocupen les variables dels nodes que el precedeixen. Aquesta *ordenació de variables* fa que els BDDs rebin la denominació de BDDs ordenats (OBDDs).

En la Figura 2-1 es pot veure un exemple d'un BDD ordenat. L'ordenació de les variables és $x_0<x_1<x_2$. Tal com es pot veure, els fills d'un node només poden ser nodes terminals o nodes no terminals controlats per una variable que ocupi una posició posterior en l'ordenament. De fet, cada node del BDD correspon a l'arrel d'un sub-BDD que representa una certa funció Booleana de les variables que ocupen una posició igual o posterior en l'ordenament global de les variables. En realitat, el succesor *F (V)* d'un node *v* correspon al cofactor de f^v respecte de $\overline{x}_i$ *(*x_i*)*. En la Figura 2-1, sigui *v* el node arrel i f^v la funció representada pel BDD. El node controlat per la variable x_1 correspon al cofactor de f^v respecte de $\overline{x}_0$, que és igual a la funció $\overline{x}_1 \cdot x_2$, que com es pot veure depèn únicament de les variables que ocupen una posició igual o posterior a x_1 en l'ordenament.

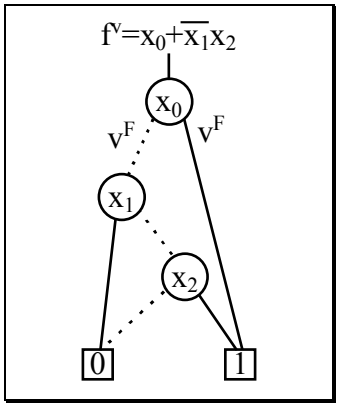

Figura 2-1 *Exemple de BDD.*

A més a més de l'ordenació de les seves variables, els BDDs es mantenen en la seva forma reduïda, que consisteix bàsicament en l'aplicació de les dues regles següents:

- Si els dos fills d'un node *v* són idèntics (apunten al mateix node, *w*), el node *v* pot ser eliminat del BDD. Els antecessors que apuntaven a *v* passaran a apuntar al seu fill, *w*.

- Si dos nodes interns del BDD *v*, *w*, representen la mateixa funció (són grafs isomorfs), aleshores només un d'ells ha de ser mantingut en el BDD. Suposem que *v* sigui mantingut. El subgraf amb arrel a *w* serà eliminat i els antecessors de *w* passaran a apuntar a *v*.

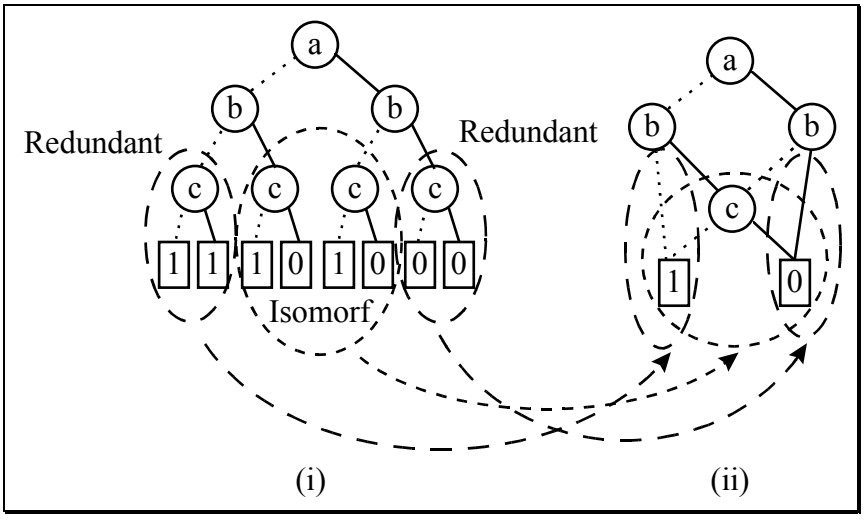

Figura 2-2 *Arbre binari no reduït (i) i el corresponent BDD reduït (ii).*

L'aplicació de les regles de reducció es pot veure en l'exemple de la Figura 2-2. En *(i)* es pot veure un arbre de decisió binària no reduït. La representació mostra nodes amb fills idèntics i també grafs isomorfs. Els nodes amb fills idèntics són redundants i poden ser eliminats de la representació, mentre que pels grafs isomorfs només cal conservar-ne un. L'aplicació de les regles esmentades dona lloc al BDD reduït que es

pot veure en *(ii)*. Tal com es pot veure la reducció suposa una disminució considerable en el número de nodes necessaris per representar la funció. L'aplicació *bddDraw* [**CRC96b**] permet visualitzar els BDDs corresponents a una suma de productes i comparar-los amb els corresponents arbres de decisió binària.

L'aplicació de les regles de reducció juntament amb el manteniment d'una ordenació fixada de les variables fa dels BDDs una representació canònica. En realitat, aquest tipus de BDDs s'anomenen BDDs reduïts i ordenats (ROBDDs), però al llarg d'aquest treball, sempre que no s'especifiqui el contrari, parlarem simplement de BDDs per referir-nos a ROBDDs.

Operació	Resultat	Complexitat temporal
Reducció	G reduït a forma canònica	$O(\|G\|\log\|G\|)$
Aplicació	$f_1 < \text{op} > f_2$	$O(\|G_1\|\cdot\|G_2\|)$
Cofactor	f_{x_i} o bé, $f_{\overline{x_i}}$	$O(\|G\|\log\|G\|)$
Composició	$f_1\big\|_{x_i = f_2}$	$O(\|G_1\|^2\cdot\|G_2\|)$

Taula 2-1 *Complexitat de les operacions entre OBDDs [**Bry86**].*

Si bé la cononicitat dels BDDs és una propietat molt interessant, sobretot per què facilita les tasques de verificació formal, segurament el fet que els BDDs s'utilitzin àmpliament com a representació de funcions Booleanes es deu a que moltes operacions fetes amb aquestes estructures tenen complexitats polinomials. En particular, qualsevol OBDD G es pot fer canònic (reduïnt-lo) amb un algoritme de complexitat $O(|G|\log|G|)$. A la Taula 2-1 es mostren les complexitats d'algunes de les operacions més habituals sobre OBDDs, suposant que G_1 i G_2 són dos OBDDs corresponents a les funcions f_1 i f_2, <*op*> és qualsevol operació Booleana binària.

La majoria de les funcions Booleanes que apareixen en els circuits lògics poden ser expressades eficientment usant un ROBDD amb un bon ordenament [**Cha93**]. Una excepció notable és la multiplicació de sencers, per la qual tots els ordenaments possibles condueixen a una complexitat exponencial de la representació respecte del número d'entrades. Malgrat això, els BDDs han pogut ser utilitzats per verificar formalment multiplicadors de 16 bits amb complexitat polinomial gràcies a la utilització de tècniques que introdueixen noves variables per reduir la complexitat de la representació [**Bur91**]. També s'han presentat funcions que tenen complexitat polinomial en una representació en suma de productes i, en canvi, complexitat exponencial al ser representades en BDDs [**Dev93**]. Cal dir, però, que aquestes funcions són l'excepció i que, de fet, la immensa majoria de funcions Booleanes presents en els circuits poden ser expressades eficientment usant BDDs.

Les tècniques per a la implementació efectiva d'un paquet de manipulació de BDDs van ser proposades en [**BRB90**]. Els BDDs es mantenen sempre en la seva forma canònica, és a dir, es treballa sempre sobre BDDs ordenats i reduïts o ROBDDs. Per tal de poder fer-ho es programa una funció ITE (*if-then-else*) que s'encarrega de dur a terme totes les possibles aplicacions entre dos BDDs i, mentre les porta a terme, empra una tècnica de *hashing* que li permet de no repetir càlculs i, el que és més important, de mantenir el BDD en una forma reduïda. En el mateix treball també es proposa l'ús d'arcs complementaris per aquelles funcions de les quals ja es disposa

de la seva forma no complementada en el BDD. De fet, només cal apuntar a la forma no complementada, indicant la necessitat d'invertir els terminals, cosa que es fa col·locant un atribut a l'arc. S'aconsegueix disminuir així la quantitat de memòria necessària per treballar amb el BDD corresponent.

Altres modificacions als BDDs originals han estat proposades en la literatura. Les més notables són la definició d'un únic espai de memòria per tots els BDDs d'una aplicació [**MIY90**], la utilització d'un tercer terminal, o una variable addicional, per representar el conjunt d'indiferents d'una funció Booleana [**ChM93**, **MIY90**], l'associació d'un cost als arcs del graf [**LPV94**] o la definició de noves regles de reducció que permeten una millor representació i manipulació de conjunts de variables o cubs [**Min93**].

Biblioteques de funcions per a BDDs

L'àmplia aplicació dels BDDs per a la representació de funcions de commutació (Booleanes) que representen el comportament d'un circuit en eines de síntesi, anàlisi i verificació de circuits digitals, ha provocat el desenvolupament de diferents paquets per la manipulació eficient de BDDs. Aquests paquets estan disponibles en l'ambient universitari en forma de llibreries de funcions, normalment programades en llenguatge C. Cada llibreria proporciona un conjunt diferent de rutines i mecanismes de manipulació dels BDDs. A part d'això, l'eficència del paquet pot ser diferent segons l'aplicació a la que vagi destinada, pel que en principi pot ser difícil saber quin paquet s'adequa millor a una aplicació concreta. En [**Cap96**] es proposa un mecanisme que permet l'encapsulació en llenguatge C++ de diversos paquets de BDDs [**Shi91**, **Long**, **Brace**, **Somenzi**] de tal manera que una aplicació pot utilitzar un o altre simplement canviant les opcions de compilació. Aquest mecanisme ha permès desenvolupar la nostra aplicació de síntesi sense preocupar-se massa dels detalls del paquet de BDDs amb el qual s'estava treballant. El paquet que s'ha utilitzat per extreure els resultats presents en aquesta memòria ha estat el desenvolupat per D.E.Long [**Long**], però en tot cas, una simple recompilació del codi permetria de treballar amb qualsevol dels altres paquets disponibles. Per a una comparació de l'eficiència dels diversos paquets de BDDs disponibles, consultar [**CRC96a**].

Xarxa Booleana: definició i modelatge d'un esquema

Una *Xarxa Booleana* $G = (V, E)$ és un graf acíclic dirigit on cada node $v \in V$ té associada una funció f^v i una variable Y^v tals que $Y^v = f^v(X,Y)$. $Y=\{Y^v\}$ és el conjunt de variables intermèdies de la Xarxa Booleana; $X=\{X^v\}$ és el conjunt d'entrades primàries (EP) i $Z \subseteq \{Y^v\}$ és el conjunt de sortides primàries (SP). Cada arc $e \in E$ és un parell ordenat de nodes (v_o, v_d), $v_o \in V$, $v_d \in V$, on v_o s'anomena el node origen i v_d el node destí.

Un *camí*, c, és una seqüència ordenada d'arcs $(e_1, e_2, \ldots, e_L)$, $e_j \in E$ que, per cada e_i, $1 < i < L$, compleix que el node origen de e_i és el mateix que el node final de e_{i-1} i el node destí de e_i és el mateix que el node origen de e_{i+1}. Direm que el node origen o inicial del camí és el node origen de e_1, i que el node destí o final del camí és el node

destí de e_L. Definirem la *longitud d'un camí c* com el número d'elements de la seqüència, és a dir, el número d'arcs que el composen.

Si es considera un arc $e \in E$, $e=(v_o, v_d)$, amb $v_o \in V$ i $v_d \in V$, v_o pertany al *ventall d'entrades* de v_d i v_d pertany al *ventall de sortida* de v_o. Es denota per V_e^v el ventall d'entrades del node *v*. De forma similar, V_s^v denotarà el ventall de sortida del node *v*. El conjunt de nodes als quals es pot arribar seguint un camí que té per origen el node *v* constitueix el *ventall transitiu de sortida* del node *v*, denotat per V_{st}^v. I el conjunt de nodes dels quals parteixen camins que tenen com a final a un node *v* forma el *ventall transitiu d'entrades* d'aquest node *v*, i es denota per V_{et}^v.

Cal indicar l'isomorfisme existent entre una Xarxa Booleana i un esquema d'un circuit a nivell de portes: cada porta present en l'esquema elèctric es correspon amb un node *v* de la xarxa isomorfa, la funció del qual, f^v, és la mateixa que realitza la porta lògica; a més, la sortida de la porta es correspon amb la variable Y^v. Per això els termes amb correspondència s'usen de manera indistinta. És a dir, el node *v* equival a la porta *v* i la sortida de la porta *v* al senyal Y^v.

Per tal de facilitar la formulació d'algorismes que actuen sobre Xarxes Booleanes, s'estableixen nivells entre els seus nodes. És a dir, es considera una *classificació topològica per nivells* dels nodes d'una Xarxa Booleana en la que cada node de la xarxa té un nivell, que es defineix a través de la següent regla recursiva:

$$\begin{aligned} N_0 &= \left\{ v \middle| v \in V, v \in X \right\} \\ N_i &= \left\{ v \middle| v \in V, \exists w \in V_e^v, w \in N_{i-1} \right\} \end{aligned} \qquad (2\text{-}2)$$

on N_i és el conjunt format per nodes del nivell *i*.

En altres paraules, el nivell d'un node és la longitud del camí més llarg entre ell i una entrada primària del circuit.

Capítol 3

Biblioteques de cel·les i Generadors de Mòduls per a Síntesi Lògica

En aquest capítol es presenta l'evolució de la metodologia de disseny al llarg de la història. Es mostra la creixent automatització del procés de disseny i la introducció de la síntesi automàtica a diferents nivells del procés de disseny dels circuits integrats. En el camp de la síntesi lògica, es presenta un estudi comparatiu de l'ús de biblioteques de cel·les bàsiques o de generadors de mòduls basats en transistors.

Introducció

A còpia de disminuir més i més la mida dels dispositius electrònics, i les dimensions dels circuits en general, s'han aconseguit reduccions espectaculars, fins fa poc temps impensables, dels aparells electrònics. Els avenços en la tecnologia permeten confinar en un espai cada cop més petit el que abans requeria enormes plaques de circuit imprès. Estimulats per les noves possibilitats que s'ofereixen s'intenten integrar més i més funcions en un sol dau de silici.

Ara bé, això té un preu: l'important increment de la complexitat del disseny i verificació d'un circuit integrat degut, en bona part, a l'aparició de tota una nova problemàtica. De l'elevada densitat de dispositius dels actuals circuits integrats es deriva tot un seguit de problemes que van augmentant la seva incidència a mida que creix el nombre d'elements per unitat d'àrea als daus de silici.

De fet l'actual problemàtica del disseny ve determinada per les línies actuals d'evolució de la tecnologia:

- constant reducció de la mida dels dispositius que es poden fabricar i reducció de les dimensions en general;
- necessitat de circuits amb freqüències de funcionament cada cop més elevades;

- aparició de noves tècniques com el *connexionat sobre les cel·les*, fruit del major nombre de capes de metall que la tecnologia posa a la nostra disposició, reduint i, fins i tot, eliminant l'àrea addicional necessària pel connexionat.

Tots aquest factors influeixen directament i de forma negativa sobre la dissipació de potència, l'electromigració, l'augment del retard degut a la reducció de les amplades de les línies de connexió, els acoblaments. Tots ells són problemes coneguts, encara que de baixa incidència sobre els dissenys amb tecnologies no submicròniques.

Segurament el problema que ha rebut mes atenció per part de la comunitat investigadora, especialment als nivells alts d'abstracció, és la dissipació de potència. La reducció de les dimensions dels dispositius implica directament la dissipació de la mateixa quantitat d'energia en un àrea menor. L'increment de la freqüència de funcionament també influeix directament sobre la dissipació del circuit. Són diversos els motius que fan de la dissipació de potència un objectiu prioritari en el disseny, al costat del retard i l'àrea. En primer lloc, l'excessiva dissipació de potència pot conduir a fallades permanents o intermitents del circuit. En segon lloc, una dissipació excessiva incrementa el cost del circuit. Efectivament, un circuit amb una alta dissipació requereix un encapsulat adequat, més car quan major sigui la quantitat de potència a dissipar. Reduir la dissipació permet reduir el cost de l'encapsulat i, per tant, abaratir el producte final. Un tercer motiu, i no per això menys important, és la necessitat de disposar d'aparells portàtils de reduïdes dimensions i de llarga operativitat. El consum determina les dimensions de les bateries necessàries i és en bona part responsable del temps que l'aparell pot operar de forma autònoma.

En termes generals, els problemes comentats ja eren de sobres coneguts, però abans només un baix nombre de dissenys topaven amb algun d'aquests entrebancs, mentre que actualment afecten de forma general a la majoria de dissenys. Així, durant molt de temps aquests problemes senzillament s'ignoraven durant el flux del disseny, tant pels dissenyadors com per les cases de desenvolupament d'eines de disseny assistit per ordinador (*CAD*). Com a resultat, es disposa de tot un seguit de complexos entorns de treball, amb un gran nombre d'eines d'ajuda per al disseny, però que, en general, no tenen en compte la nova problemàtica apareguda, especialment, arrel del desenvolupament de dissenys submicrònics.

Per arribar a solucions sistemàtiques dels problemes plantejats, caldrà integrar el seu tractament en el flux de disseny. És imprescindible tenir en compte les restriccions durant tot el procés de disseny. L'aplicació del conjunt de tècniques que fan que cada una de les tasques involucrades en el procés de disseny tinguin en compte les prestacions finals que se li demanaran al circuit i que treballin en la direcció adequada per aconseguir-les, donen lloc al disseny guiat per prestacions. Aquestes tècniques han demostrat aconseguir millores importants respecte a les alternatives clàssiques.

Les noves exigències, tant de disseny com tecnològiques, requereixen noves eines que facilitin les tasques de concepció i implementació d'un circuit integrat. Alguns canvis ja s'han incorporat al procés de disseny, substituint en alguns casos les tècniques tradicionals. Així, per exemple, les descripcions amb esquemes elèctrics han estat substituïdes per les descripcions del circuit en base a llenguatges de

descripció de *hardware* (*HDLs*), introduint una major flexibilitat respecte a la tradicional entrada d'esquemàtics.

La descripció d'un circuit mitjançant un HDL implica el posterior procés de síntesi automàtica d'aquest circuit. Al procés de síntesi, i en especial a la síntesi lògica, se li han incorporat amb èxit les tècniques d'optimització guiades per prestacions. Així, el procés de síntesi treballa per aconseguir complir amb les restriccions imposades sobre el disseny.

En aquest capítol es revisarà breument l'evolució històrica de les metodologies de disseny de circuits integrats, des dels seus inicis fins als nostres dies, fent èmfasi especialment en la creixent automatització del procés de disseny. L'automatització ha portat a la definició de metodologies de disseny basades en biblioteques de cel·les estàndard, que seran breument comentades. A continuació es descriurà la problemàtica associada al desenvolupament d'aquestes biblioteques i els treballs en el camp de l'automatització del disseny i caracterització de les cel·les que la formen. Finalment, s'introduirà una nova metodologia de disseny, on les biblioteques de cel·les no seran necessàries i es substituiran per un generador de mòduls guiat per prestacions. Els avantatges que aquesta nova metodologia suposa en el camp de la síntesi lògica seran comentats i se n'extrauran les conclusions pertinents.

Metodologies de disseny i dependència tecnològica

En aquest apartat es pretén fer una revisió de les principals alternatives d'implementació basades en un conjunt de cel·les bàsiques, que es prenen com a base, donada una certa tecnologia i estratègia de disseny, per tal d'analitzar les prestacions que es desprenen del seu ús, pensant principalment en la seva integració en el procés de síntesi automàtica.

No es considerarà el cas dels compiladors d'estructures (PLA, RAM, ROM, multiplicadors), que representen l'opció més eficient d'implementació tecnològica, ja que estan lligats a una elevada regularitat i simetria i precisament per això requereixen unes restriccions estructurals importants. Com a conseqüència presenten una menor flexibilitat en termes de funcionalitat i de compromís entre prestacions d'àrea, velocitat i consum de potència.

Tampoc pretenem considerar alternatives com les dels dispositius configurables tipus FPGA o EPLD, tot i que una part important dels treballs en el camp del mapatge tecnològic s'enfoquen a aquests tipus de dispositius [**EM91**, **FRV91**, **Kar91a**, **Kar91b**, **CR92**, **MBS92**, **CD93**, **HSA93**, **KP93**, **ST93**, **HOIW94**, **CWL96**]. La raó d'aquesta renuncia es basa en la gran diferència entre els dos problemes. Aquest tipus de dispositius presenten igualment una molt alta regularitat estructural, ja que es basen en una cel·la de base única, el bloc lògic combinacional o una memòria programable capaç de realitzar qualsevol funció d'un número reduit de variables. El problema de concepció es trasllada a un problema de descomposició, assignació i connexió, perdent una part de la flexibilitat existent en la síntesi lògica basada en biblioteques de cel·les estàndard o en generadors de mòduls. Per tant, els compromisos entre diferents objectius de disseny (àrea, velocitat i consum) presenten una "discretització" important en l'espai de solucions.

Evolució històrica

L'evolució en la metodologia de concepció de circuits integrats ha estat constant des dels mateixos inicis d'aquests, l'any 1959 gràcies als desenvolupaments de J.Kilby i R.Noyce. Aquesta evolució ha estat possible mercès a la coincidència de l'evolució tecnològica i la de les plataformes *hardware* i eines de disseny assistit per computador [**NS87**, **Sah88**, **She93**, **Ser94**, **Pel95**].

Els principals elements que han dirigit aquesta evolució són: el cost econòmic, tant de desenvolupament com de fabricació; el nivell d'integració i les prestacions de velocitat i consum; els temps de disseny i fabricació, etc. La interacció entre aspectes econòmics i tecnològics ha estat difícil de modelar, de manera que s'ha tendit a analitzar-los per separat. Es aquí on trobem les típiques corbes de cost en funció de la tecnologia utilitzada i el número de circuits a produir.

D'altra banda, a nivell de prestacions tecnològiques, s´han desenvolupat uns models analítics suficientment bons per a les diferents tecnologies. Al seu torn, aquests models han estat automatitzats en eines de concepció. Aquestes eines han seguit una vertiginosa evolució cap a nivells d'abstracció cada cop més elevats en tant que això permet acostar-nos més a una especificació funcional dels circuits i sistemes electrònics i, per tant, a les aplicacions industrials d'aquests.

En aquesta evolució és on la necessitat d'una metodologia clara, senzilla i eficient ha estat més necessària. Això es reflexa tant en l'estructura de les eines de concepció com en l'accés als processos de fabricació. Les companyies que exploten comercialment aquests darrers, han tendit a dividir el procés de disseny en etapes, per tal de minimitzar la dispersió de valors de prestacions en el pas que porta de les especificacions d'alt nivell fins a les màscares utilitzades durant el procés de fabricació. Es aquí on ha pres el seu sentit el concepte de biblioteques de cel·les bàsiques, que permeten un bon compromís entre la necessitat de verificar la correcció dels circuits per part dels fabricants abans d'entregar-los al client, i la necessitat de formalitzar les especificacions que han de complir [**ESS10**, **ESS07**, **MIE05**].

En la dècada dels 80 van aparèixer les biblioteques de cel·les. Es basaven en les descripcions funcionals de les cel·les a l'estil de les existents per als nivells d'integració baixos i mitjos (*LSI* i *MSI*). Les cel·les havien de presentar unes característiques estructurals genèriques similars, tant a nivell elèctric com geomètric, per tal de facilitar la seva posterior connectivitat a nivell de circuit integrat [**Smi89**].

L'avantatge fonamental per a l'usuari, residia en el fet que no calia abordar el disseny geomètric del circuit integrat (a nivell de màscares) sinó que aquest procés es podia fer automàticament, mitjançant una sèrie d'eines que permetien tant generar com configurar i/o connectar els elements bàsics o cel·les de la biblioteca [**Gaj88**].

Addicionalment, a aquestes cel·les se'ls podia associar una sèrie de paràmetres per tal d'estimar les seves prestacions bàsiques en termes de superfície i velocitat. Es facilitava així una estimació global de les prestacions del circuit a un nivell d'abstracció més elevat. En aquesta època, les restriccions de dissipació de potència eren encara molt relaxades, excepte per a aplicacions especialitzades, degut a l'encara relativament baix nivell d'integració (processos per sobre de la micra). Però aquestes

estimacions a alt nivell han anat perdent precisió a mida que ha augmentat l'escala d'integració, donat que els efectes paràsits lligats al connexionat s´han anat fent cada cop més importants respecte als paràmetres de la pròpia cel·la [**Bak90**, **SY93**, **COM95b**]. Al mateix temps, la degradació de prestacions produïda per l'escalfament degut a la dissipació de potencia d'un número més elevat de transistors, ha portat a la necessitat de l'aplicació d'una sèrie de tècniques que permetin estimar millor el consum [**DHNTB95**, **Mus96**]. Les eines capaces d'estimar eficientment el consum, però, estan encara en fase de recerca activa i desenvolupament.

Seguint l'evolució tecnològica podem fer-nos una idea més clara de com ha evolucionat l'estimació d'aquestes prestacions. Els processos més populars en la dècada dels 60 i 70 es basaven en transistors bipolars. La quantitat de dispositius dedicats a implementar una funció elemental era elevat, en tant que eren necessaris tota una sèrie de circuits que permetien polaritzar els transistors. La correlació entre funcionalitat o número d'entrades d'una cel·la respecte de la seva ocupació en superfície era força baixa. El control de la funció lògica a través del corrent de base dels transistors bipolars (efecte tridimensional) feia que aquests transistors fossin molt poc escalables Calia afegir un número sencer de dispositius en cas de necessitats de corrent majors. Aquest fet, juntament amb la limitada connectivitat, va donar pas a enunciats clàssics de restriccions com ara el de màxim ventall de sortida, referent al número màxim d'entrades connectables a una sortida donada. Aquestes restriccions evitaven que els nivells de tensió a la sortida es degradessin, fet que provocaria la pèrdua de la funcionalitat. Estructures d'aquest tipus són encara utilitzades avui en dia, tot i que d'una forma molt més selectiva, en processos BiCMOS, com a *buffers* amb una circuiteria de polarització molt menor, o en técniques d'alta velocitat (*ECL o CML*) molt especialitzades [**PWH89**, **Wil90**, **EBE93**].

El següent procés tecnològic que es va popularitzar va ser el procés NMOS (o PMOS). Es caracteritzava perquè d'una banda desapareixia tota la circuiteria de polarització, que era substituïda per un únic transistor de càrrega, estructuralment igual que els de control lògic, però de característiques elèctriques diferents. D'altra banda, els dispositius tenien el comportament bidimensional característic dels canals en estructures MOS i eren, per tant, dimensionables. Com a conseqüència, presentaven valors característics diferents dins del circuit integrat. Per exemple la capacitat d'entrada del transistor variava segons la dimensió d'aquest. El concepte de màxim ventall de sortida va passar, excepte en els casos en que tots els transistors del circuit eren iguals per construcció, de valors sencers (indicant el número de transistors d'igual dimensió que es podien carregar) a valors reals (indicant directament la capacitat màxima que es podia carregar) [**MC80**]. Això permetia que les estimacions de superfície fossin més susceptibles de ser extretes a partir de la funcionalitat: una porta NMOS que implementés una funció lògica negada simple de n variables d'entrada contenia $n+1$ transistors, i la superfície ocupada era proporcional a la suma de les dimensions dels transistors. Tot i això, es mantenien encara un conjunt de restriccions referents al dimensionament dels transistors de càrrega i de control, o funcionals, que asseguraven el funcionament correcte de cadascuna de les portes lògiques.

Tant el procés bipolar com el procés NMOS (o PMOS), presenten un consum de potència estàtic degut a que utilitzen circuiteria de polarització fixa. Aquest fet, poc

important per a escales d'integració reduïdes, fou un altre dels que van determinar la següent evolució tecnològica. A la dècada dels 80 es popularitzen els processos CMOS que encara avui corresponen a la majoria dels processos utilitzats industrialment. Aquests processos presenten l'avantatge que és possible construir portes lògiques, la funcionalitat de les quals és independent de les dimensions dels transistors que les formen. En l'anomenada lògica complementària, la més utilitzada, cada porta CMOS conté un bloc format per transistors PMOS que connecta la tensió d'alimentació amb la sortida, i que es complementari d'un altre bloc format per transistors NMOS que connecten la sortida a la tensió del substracte [**GD85**, **WE93**].

Utilitzant processos CMOS, han estat també popularitzades moltes altres estructures de transistors que permeten formar portes lògiques. La lògica pseudo-NMOS permet estalviar el bloc de transistors PMOS, posant-ne un de sol com a càrrega. Aquesta lògica s'ha anat perdent a mesura que el nivell d'integració ha augmentat degut a l'elevat consum de potència estàtic. També són possibles en processos CMOS les lògiques dinàmiques, en les quals els blocs de lògica combinacional estan governats per senyals de control (rellotges o fases de rellotge). Permeten velocitats de procés més elevades, però per contra es comporten pitjor respecte de l'automatització dels processos de connexionat, principalment degut a que els senyals de control esdevenen crítics. Tot i això, avui dia encara suposen una alternativa vàlida per a la concepció de circuits de molt alta velocitat, com per exemple les lògiques de tipus TSPC [**AS90**].

Les biblioteques de cel·les.

Els elements fonamentals de relació entre procés tecnològic i procés de disseny són: les *regles de disseny*, que defineixen els límits geomètrics de les diferents capes de materials presents en un CI; els *paràmetres elèctrics* dels dispositius presents en un CI, tant els corresponents a elements passius (capacitats i resistències) com a components actius (MOS i unions PN); finalment, existeixen altres paràmetres addicionals com els criteris per a evitar l'efecte *latch-up*, els límits de corrent, l'estructura dels PADS, les dimensions màximes dels circuits, etc…

A partir d'aquests elements és possible el disseny de circuits integrats a nivell de transistor, mercès a tècniques de disseny completament a mida (*full-custom*). Aquestes tècniques permeten una òptima relació entre prestacions elèctriques i superfície ocupada. Malgrat això, aquesta tècnica de disseny no és la més utilitzada degut a que requereix un elevat temps de desenvolupament.

Les tècniques de disseny de circuits integrats parcialment a mida (*semi-custom*) es basen en la utilització de biblioteques de cel·les bàsiques i blocs parametritzables. Les cel·les estan estretament relacionades tant amb un procés tecnològic com amb uns criteris determinats, escollits pels responsables de la biblioteca en funció de l'orientació d'aquesta en termes d'aplicacions [**CT90**]. Aquests criteris són diversos i fan referència als següents aspectes:

- Criteris de disseny: ocupació de superfície, característiques temporals i dissipació de potència.

- Criteris de fabricació: temps de fabricació, en funció del número de màscares a processar.

- Criteris de biblioteca: número de cel·les, temps de disseny i caracterització, rang de treball de les cel·les, i facilitat de manteniment que permeti seguir l'evolució tecnològica.

Les dues estratègies fonamentals que reflecteixen criteris extrems, són:

***Sea-of-Gates* (o *Gate Array*) [**Pes93**].**

En aquesta metodologia de disseny, els nivells d'àrea activa, implantacions i polisilici, estan ja definits en un circuit integrat predifós. El circuit conté parells de transistors (NMOS, PMOS) de dimensions fixes, tot i que existeixen vàries dimensions per possibilitar diferents necessitats d'ocupació. La formació d'estructures lògiques es realitza per la personalització del connexionat dels transistors, o parells de transistors, mitjançant el processament de les darreres capes conductores, típicament dos nivells de metall.

Tant el temps de processament d'un circuit integrat específic com el de desenvolupament d'un biblioteca de cel·les són reduïts. Les prestacions obtingudes solen ser baixes comparades amb altres estratègies. És difícil obtenir una bona ocupació de superfície, i les prestacions elèctriques presenten problemes tant d'ordre dinàmic (asimetria important dels temps de propagació en funció de la porta lògica) com estàtic (marges de soroll reduïts). Les prestacions elèctriques es poden millorar utilitzant transistors en paral·lel, la qual cosa ens porta a un comportament del tipus *ventall de sortida discret*, com el comentat anteriorment per als circuits amb transistors bipolars.

L'estimació de prestacions d'àrea velocitat i consum de potència en aquestes estructures no és difícil, ja que existeix un nombre reduït de parells de transistors amb dimensions diferents. Però donat que les cel·les no estan precaracteritzades, cal determinar un conjunt paràmetres corresponents als macromodels de les cel·les, que permetin realitzar simulacions temporals amb un grau de precisió acceptable, principalment per a tecnologies submicròniques. En alguns casos aquest procés es substitueix per un procés de *back-annotation*, on la informació extreta de les màscares del circuit es realimenta cap als esquemes elèctrics.

***Cel·les Estàndard* [**KG85**, Hei86**].**

Les cel·les estàndard en contrast amb la metodologia anterior, requereixen el processament de totes les capes per a cada disseny. El temps de desenvolupament de la biblioteca de cel·les serà també més elevat, ja que s'optimitzen les prestacions per a cadascuna de les cel·les. Com a contrapartida, tant les característiques elèctriques, com les d'ocupació de superfície son millors que en el cas dels *Sea-of-Gates* degut a que s'utilitza únicament la superfície imprescindible.

> L'estratègia d'ús de cel·les estàndard permet la integració d'altres tipus de blocs, l'ús dels quals s'ha incrementat amb l'augment de la densitat d'integració, com són les estructures regulars generades amb compiladors d'estructura (PLA, RAM, ROM, multiplicadors, *data-path bit-slice*,...), cel·les i estructures analògiques i, més recentment, sensors i actuadors.

Altres estratègies possibles per a biblioteques de cel·les (*optimized array* [**S1400**], *gate matrix*,...) són menys populars, ja que presenten els problemes de les dues anteriors, és a dir, dimensionament difícil degut a l'ús de transistors de dimensions fixes i, al mateix temps, necessiten del processament de totes les capes [**WE93**].

Els criteris de disseny de biblioteques orientats a aplicacions concretes, solen venir condicionats per l'optimització de requeriments específics, com per exemple tensió d'alimentació reduïda, velocitat molt elevada, consum mínim, etc. Aquests diferents criteris proposen l'existència, per a un mateix procés tecnològic, de diverses biblioteques de cel·les estàndard, que poden fins i tot conviure en un mateix circuit integrat [**Sun87**, **Pig94**].

També cal remarcar que l'evolució cap a l'automatització del procés de disseny ens porta a l'aparició de representacions més flexibles per a les cel·les d'una biblioteca. Cal considerar aspectes més abstractes que un esquema elèctric dimensionat o una geometria fixa. Es pretén disposar d'unes estructures de base independents no només del procés tecnològic, sinó de les dimensions dels dispositius. Aquestes, s'introduiran en funció del tipus de biblioteca i, així, es generaran cel·les específiques després de processos de síntesis i compactació. Aquest procés es pot realitzar per a qualsevol de les cel·les de la biblioteca, tant combinacionals com seqüencials, tot i que en aquest darrer cas el número de paràmetres temporals involucrats en una optimització és més elevat [**BMK87**, **MDB87**, **Mas91**, **DNS94**, **COM95a**].

La tendència oposada a la diversificació de cel·les proposa que les biblioteques redueixin el número de cel·les elementals que les formen, com en tenim exemples en les biblioteques procedents de diferents fabricants. Cel·les més complexes es poden implementar mitjançant combinacions de cel·les simples connectades amb els mateixos mètodes de posicionament i connexionat utilitzats per a la realització del circuit (*soft-cells*). Aquesta tècnica permet la convivència de diferents alternatives de disseny d'un mateix bloc funcional per tal d'adaptar-lo a requeriments diversos. Per exemple, poden conviure diferents arquitectures per a sumadors, multiplicadors, etc… [**ESS10**, **ESS07**, **MIE05**]

Tot seguit s'analitzaran els aspectes elèctrics i geomètrics involucrats en el desenvolupament d'una biblioteca de cel·les estàndard.

Aspectes elèctrics

La determinació dels paràmetres elèctrics globals per als models de simulació és potser el pas més complicat del procés de disseny d'una biblioteca de cel·les estàndard, ja que involucra el compromís global de característiques que posteriorment s'optimitzaran localment en cada cel·la [**CT90**, **LMK90**].

La definició de l'entorn de treball característic de les cel·les, establirà els valors dels paràmetres elèctrics derivats de les condicions d'entorn. Aquestes seran, fonamentalment, el màxim ventall de sortida, en termes de *capacitat màxima* permesa a la sortida d'una cel·la, i el *temps de transició màxim* per a les entrades i sortides, ja que els temps de propagació dels senyals depenen d'aquests per complir amb les prestacions temporals anunciades en el corresponent full de característiques [**BCA88**].

Un cop definits els paràmetres característics, la realització de cadascuna de les cel·les es veurà afectada de manera diferent en termes de necessitat de *bufferització* a les sortides o de reducció de capacitats a les entrades, la qual cosa dirigirà l'elecció de l'estructura interna de la cel·la i les seves prestacions finals tant de velocitat com de consum. Seguint el raonament proposat anteriorment es clar que diferents condicions d'entorn afectaran cada cel·la de manera diferent [**MAA94**].

La conclusió és, per tant, que a aquest nivell no es pot parlar d'un escalat de les característiques de velocitat en funció de paràmetres funcionals de la cel·la.

A nivell de circuit, tenim com a element addicional el grau de flexibilitat en els marges de tolerància de les cel·les respecte de variacions externes, com les de procés tecnològic, temperatura o tensió d'alimentació. Aquests marges determinaran el compromís entre la proporció de circuits acceptats com a correctes pel fabricant, i l'esforç del dissenyador per a complir les especificacions, tenint en compte les variacions en la velocitat de funcionament i consum d'un circuit.

Aspectes geomètrics

En primer lloc, l'optimització en superfície és altament depenent del dissenyador que realitza les cel·les. En segon lloc, el grau de compactació depèn de les dimensions dels transistors, però l'altura de les cel·les de la biblioteca normalment està fixada. Sovint es pot mantenir una mateixa disposició dels transistors quan canvien les condicions externes, la capacitat a la sortida o el temps de transició, degut per exemple a un entorn d'aplicació diferent (reducció de la tensió d´alimentació) o a una evolució en la tecnologia (escalat de dimensions). Aquest fet implica, però, que en certes ocasions l'altura de la cel·la no és completament aprofitada per la funcionalitat de la cel·la.

La conclusió a la qual s'arriba és, per tant, que la correspondència entre els paràmetres d'alt nivell d'una cel·la, com poden ser número de transistors, número d'entrades, funcionalitat, o altres, que pot tenir sentit per tal de fer estimacions a nivell global, pot presentar una gran dispersió quan s'analitzen les cel·les d'una determinada biblioteca. Com a conseqüència, la selecció òptima d'una d'aquestes cel·les per a una determinada funció caldrà avaluar-la amb molta més cura que quan es consideren metodologies on existeix una bona correlació entre els paràmetres d'alt nivell i els valors reals de superfície ocupada, com passa amb els generadors de mòduls.

CLIP: desenvolupament de biblioteques de cel·les CMOS

Actualment, els sistemes *CAD* permeten reduir dràsticament el temps de posta a punt d'una biblioteca de cel·les per a una nova tecnologia, tot i que són poques les eines orientades específicament a aquesta tasca. Per tal d'omplir aquest buit vam proposar el desenvolupament de l'eina *CLIP* que incorpora, dins un entorn *CAD* per a disseny *VLSI*, rutines de generació automàtica de màscares i de caracterització elèctrica orientades específicament a la generació de biblioteques de cel·les estàndard [**RRPCT91**, **RRPCT91a**, **RRPCT91b**, **RRPCT92**, **RRPSCT93**].

L'entorn *CLIP* ha estat concebut per a proporcionar al dissenyador les eines suficients per a fer possible l'obtenció automàtica de les diferents representacions de cada cel·la.

Per tal d'aconseguir aquest objectiu, *CLIP* utilitza una biblioteca de base, en la qual s'inclouen les representacions independents de la tecnologia, com ara els símbols, les descripcions funcionals, els esquemes elèctrics amb els transistors sense dimensionar, i les representacions geomètriques a nivell simbòlic. A partir d'aquí, cal establir els paràmetres globals associats a la biblioteca com ara temps de transició màxim la capacitat màxima a la sortida, i dimensionar correctament l'esquema elèctric de cada cel·la.

Amb els esquemes elèctrics obtinguts en l'etapa anterior, les regles geomètriques de disseny adequades al procés tecnològic, i alguns paràmetres addicionals, el generador de màscares de *CLIP* facilita l'obtenció de la representació geomètrica de cadascuna de les cel·les de la biblioteca. En aquest pas es contemplen diversos procediments per a la generació. Per a cadascun d'ells s'ha desenvolupat l'eina adequada que facilita la tasca del dissenyador, essent fins i tot possible la generació automàtica sense la intervenció del dissenyador.

Per a l'obtenció dels fulls de característiques de cada cel·la, *CLIP* incorpora un entorn, anomenat genèricament **CIRCUS** (*CIRcuit electrical Characterization Using SPICE-like simulators*), que permet obtenir automàticament les característiques temporals de cadascuna de les cel·les per a un entorn de caracterització donat [**RRPCT91a**, **RRPSCT92a**, **RRSPCT93**, **RRPSCT93a**]. D'aquesta forma, es disposa dels paràmetres elèctrics i temporals necessaris per a la confecció dels corresponents fulls de dades, els quals, al seu torn, seran utilitzats per a la preparació dels models de simulació a nivell lògic.

El generador de màscares

El generador de màscares ha estat desenvolupat amb vistes a adaptar-se a canvis tecnològics raonables [**Rie91**]. Per tant, pot considerar-se independent de la tecnologia [**Pig90**, **TVT86**]. De fet, la descripció tecnològica, a nivell de regles de disseny geomètric, és proporcionada juntament a altres paràmetres addicionals a l'eina, anomenada *GenLIB*, que ha estat creada amb la finalitat d'automatitzar les decisions referents a la topologia bàsica de les cel·les i també per a generar els fitxers adequats a les diverses metodologies de disseny contemplades. De manera que un canvi tecnològic pot significar, a nivell geomètric, simplement modificar el fitxer d'entrada a *GenLIB* i compilar de nou les cel·les a nivell simbòlic per tal d'obtenir les corresponents geometries.

GenLIB genera la topologia bàsica de les cel·les calculant:

- la distància entre alimentació i terra,
- l'amplada del pou N,
- l'amplada de les línies d'alimentació i terra,
- les restriccions per a la connexió lateral de cel·les.

Considerant una biblioteca en la qual totes les cel·les tinguin idèntica altura, a partir de les regles de disseny geomètric s'estima la mínima altura requerida per a implementar les diferents funcions presents en la biblioteca. Bàsicament, es divideix la cel·la en zones diferents: la corresponent a les línies d'alimentació, la mínima altura requerida per a implementar un transistor de cada tipus, i la destinada a permetre el pas de línies internes de connexió, o zona de canal.

Estimant el consum d'una fila de cel·les, d'una longitud fixada per la tecnologia, i sabent la intensitat màxima per secció de metall, es determina l'amplada de les línies d'alimentació.

L'àrea requerida per a implementar els transistors **P** i **N** s'obté a partir de normes tecnològiques, considerant transistors de dimensions mínimes que, de fet, poden créixer cap a l'interior de la zona de canal si és necessari.

Pel que fa a la zona de canal, cal esmentar la seva dependència bàsica, a part de les normes de disseny, del número de línies de comunicació interna permeses. Aquest número ha de ser decidit en base a la complexitat de les cel·les a implementar.

Un cop definida la topologia bàsica de les cel·les, l'obtenció de les màscares de les mateixes, dins *CLIP*, pot realitzar-se utilitzant diverses metodologies de disseny:

i) ***Disseny a mida***. El dissenyador utilitza un editor de geometria per a dibuixar les màscares. Aquest és el mètode tradicional i el que millors resultats proporciona, tot i que és costós en temps. Si s'utilitza aquesta metodologia de disseny, *CLIP* afegeix les restriccions topològiques, que totes les cel·les han de complir, als fitxers de verificació de regles de disseny proporcionats pel fabricant, de manera que puguin ser comprovades pel programa de verificació de regles de disseny durant el desenvolupament de la biblioteca. La verificació assegura unes màscares lliures d'errors de disseny.

ii) **Generació automàtica**. Aquesta metodologia utilitza un generador de *màscares* a nivell simbòlic per a obtenir, a partir d'un esquema elèctric a nivell de transistors, la geometria de les cel·les. *GenLIB* proporciona els fitxers tecnològics per al generador. Per tant, aquesta metodologia és, potser, la més atractiva, ja que les màscares es generen de forma totalment automàtica, i de forma molt ràpida. Malgrat això, el resultat final pot no ser excessivament bo comparat amb una versió feta a mida per un dissenyador experimentat.

iii) **Generació semiautomàtica**. En aquest cas, s'utilitza el generador de màscares a nivell simbòlic per obtenir, en pocs minuts, una primera versió de les màscares de la cel·la. Seguidament, el dissenyador aplica tècniques de dissenya a mida (simetries, rotacions, desplaçaments verticals,...) a nivell de simbòlic, per tal de millorar els resultats. Aquesta metodologia aprofita la

rapidesa del generador i l'experiència del dissenyador, aconseguint un bon compromís entre temps de disseny i prestacions del resultat final.

Síntesi lògica orientada a generadors de mòduls

El procés de síntesi automàtica consisteix en un refinament successiu d'una descripció d'alt nivell d'un circuit [**CG92**]. Tradicionalment, l'última etapa d'aquest procés ha estat la síntesi lògica combinacional multi-nivell. Aquesta s'ha vingut realitzant en dues fases [**BHS90**]. La primera consisteix en la reestructuració, minimització i optimització d'una Xarxa Booleana que representa el circuit. L'objectiu d'aquesta etapa, independent de la tecnologia, consisteix en extreure funcions comunes en el circuit per tal d'estalviar àrea. La segona fase de la síntesi lògica és el mapatge tecnològic, que consisteix en generar la implementació de la Xarxa Booleana ja optimitzada. El mapatge tecnològic depèn clarament de la tecnologia, i habitualment consisteix en implementar el circuit usant únicament les cel·les que s'ofereixen en una determinada biblioteca. El circuit resultant ha de complir amb les restriccions imposades al disseny original. Inicialment, la restricció principal era l'àrea [**Keu87**, **DGRSW87**, **Ber88**, **MM90**, **KDN92**]. Posteriorment s'hi va introduir el retard [**BCGH86**, **MJH89**, **MM93**, **CP95**] i actualment també la dissipació de potència [**TAM93**, **TPD93**].

Les tècniques de mapatge tecnològic referenciades usen una biblioteca de cel·les concreta per fer el trasllat de la Xarxa Booleana independent de tecnologia al circuit final en la tecnologia escollida. Aquest, s'ha d'implementar combinant les funcions de les quals es disposa en la biblioteca de cel·les, complint amb els condicionaments del disseny. Aquest esquema, que podríem anomenar clàssic, comporta dos inconvenients:

- És necessari el *manteniment d'una biblioteca* de cel·les prou extensa per poder disposar d'un cert grau de flexibilitat durant el procés de síntesi.

- La *limitada diversitat de funcions* i el relativament *reduït nombre de diferents implementacions per cada funció* de les quals es disposa en una biblioteca de cel·les, limita el rendiment de la síntesi lògica. Quan major és el nombre de funcions a la biblioteca i major sigui el nombre de diferents implementacions en base als diferents objectius de disseny (bàsicament compromisos àrea-retard-consum) major serà la flexibilitat del procés de síntesi.

La generalització d'aquest postulat condueix a la utilització d'eines per a la síntesi automàtica a nivell geomètric: els generadors de mòduls. Aquestes eines tenen com a objectiu generar les màscares corresponents a la implementació de qualsevol funció, complint amb les especificacions. Els detractors de la generació automàtica de màscares, argumenten l'existència de diversos inconvenients a l'hora de treballar amb aquestes eines:

- Les *dimensions d'un mòdul generat són més grans* que les que es poden obtenir per la mateixa funció amb un disseny manual fet per un expert en la matèria.

Aquí és necessari fer dues remarques. D'una banda, aquestes diferències en àrea s'estan reduint gràcies a l'aparició d'algoritmes cada cop més eficients. Per altra banda, la generació de mòduls obté un guany en àrea pel fet que disposa de funcions complexes que, senzillament, no existeixen a les biblioteques de cel·les estàndard. El conjunt de cel·les necessàries per realitzar la mateixa funció, ocupa un àrea superior al mòdul generat.

- La *verificació del mòdul generat* és difícil.

  El problema fa referència a la dificultat de trobar models prou acurats. L'ús de models reals no és adequat donat el llarg temps de simulació necessari. En general, s'utilitzen models RC distribuïts que permeten una velocitat de simulació de varis ordres de magnitud superior. Les primeres estimacions basades en aquests models presentaven errors relativament elevats, al voltant del 30%. Recentment han aparegut models RC que aconsegueixen un grau de precisió molt elevat (menys d'un 5% d'error), conservant els avantatges en temps de càlcul [**Pes93**].

- Els generadors de mòduls *no són totalment independents de la tecnologia*.

  Els canvis més habituals en les tecnologies corresponen a escalats. En aquests casos no cal fer cap modificació a les eines de generació. Simplement cal modificar el fitxer tecnològic. Només canvis en el número o tipus de màscares requeriran un cert ajust en les eines de generació.

Malgrat els comentaris anteriors, l'ús de tècniques de generació automàtica de mòduls presenta nombrosos i importants avantatges:

i) S'elimina la necessitat del *desenvolupament i posterior manteniment de les biblioteques de cel·les estàndard*. Qualsevol canvi en la tecnologia implica l'actualització de les cel·les de les biblioteques associades al procés. Per una eina de generació automàtica de mòduls correctament parametritzada un canvi tecnològic només suposarà carregar un nou fitxer tecnològic.

ii) Es *Disposa de totes les funcions* realitzables en la tecnologia considerada, a més d'una *implementació a mida per cada porta del circuit* que permet ajustar al màxim el compromís àrea-retard-consum, reduint la necessitat d'iteracions sobre el procés de síntesi. La utilització de funcions complexes comporta un benefici addicional. El fet d'introduir funcions complexes sobre els camins no crítics farà que aquests augmentin el seu retard, fent que la diferència de retards entre els diferents camins del circuit sigui menor. D'aquesta forma es redueix el número de polsos espuris, reduint de forma important el consum.

iii) El procés de generació *aprofita la informació* disponible durant la síntesi lògica per dur a terme la implementació del mòdul. Com a exemple, l'anàlisi de l'activitat de commutació dels nodes del circuit realitzada durant la síntesi lògica es pot utilitzar per dirigir les etapes de posicionament, connexionat i optimització.

És necessari que el procés de disseny que hi ha rera la síntesi treballi en la mateixa direcció que aquesta per aconseguir complir amb tot el ventall de restriccions i condicionaments del disseny. Donat que la mesura de determinats paràmetres (com ara el consum) és resultat d'una estimació, si el posterior procés d'implementació del disseny no té en compte les restriccions, és molt probable que no arribi a un circuit adequat. La implementació del resultat de la síntesi ha de estar dirigida per les restriccions imposades al disseny i ha d'afinar el mòdul final per ajustar aquells paràmetres que presenten una major variació respecte al que s'ha pogut estimar en la síntesi lògica, fent ús de la informació més precisa de la qual es disposa durant aquesta etapa.

El més ampli ventall de possibilitats d'implementació de cadascuna de les funcions presents en un circuit i la millora en les eines de generació de mòduls guiats per prestacions permeten augurar un increment notable en l'ús d'aquest tipus de tècniques. A continuació es descriurà el funcionament general d'un generador de mòduls guiat per prestacions.

Esquema general d'un generador de mòduls

Tradicionalment, els generadors de mòduls guiats per prestacions descrits a la literatura [**Chen88**, **Tsa88**, **Mor93**, **Lin94**] estan basats en la repetició d'un procés iteratiu.

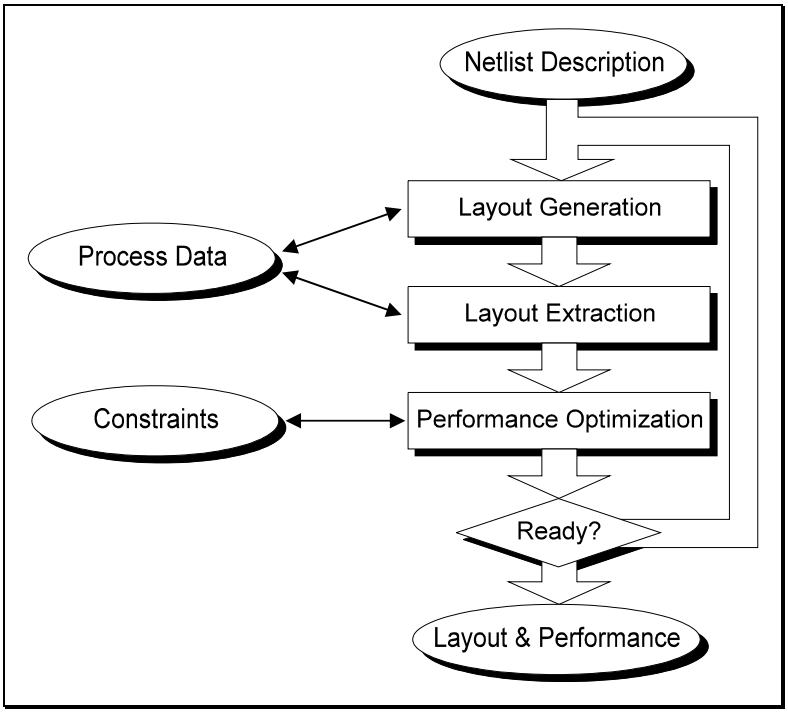

Figura 3-1 *Esquema clàssic d'un generador de mòduls guiat per prestacions.*

Tal com es mostra a la Figura 3-1, s'obté un circuit que compleix les restriccions a partir d'una descripció a nivell de transistors i seguint un procés iteratiu. Aquest procés involucra tres tasques principals:

- generació del mòdul o circuit sense tenir en compte les restriccions;
- extracció de capacitats paràsites;
- optimització del rendiment.

Aquest últim procés canvia el valor de les capacitats paràsites especialment degut al canvi en les dimensions dels transistors. Això pot portar a transistors excessivament grans per a l'àrea de la qual es disposa, violant regles de disseny i fent necessari un nou procés de generació del mòdul que haurà de tenir en compte els paràmetres obtinguts a l'etapa d'optimització. Així, de forma iterativa, ens acostarem al que serà el circuit final que complirà amb les restriccions imposades sobre el disseny.

Aquest esquema de generació de mòduls presenta l'inconvenient de no poder predir el nombre d'iteracions necessàries per assolir un circuit adequat. A més, la filosofia del procés és la de refinar un primer mòdul proposat sense tenir en compte restriccions per fer que s'acomodi a les necessitats exigides.

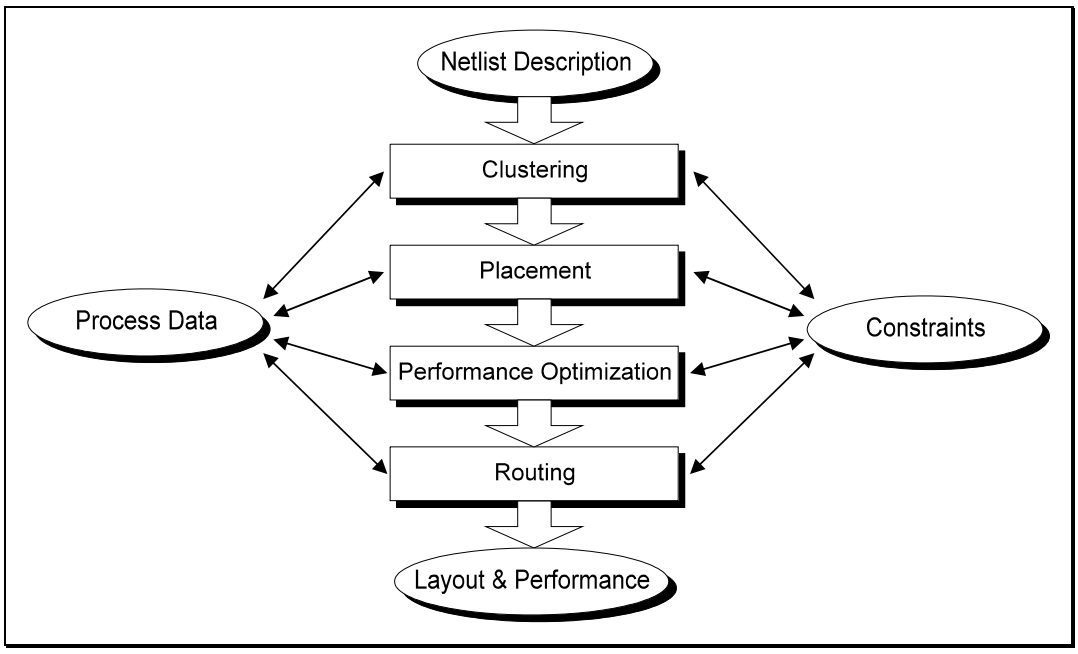

Figura 3-2 *Esquema d'un generador de mòduls guiat per prestacions sense iteració*

A [**Pes94**, **VMRPC96**] es presenta una alternativa a l'esquema iteratiu tradicional. L'estructura proposada es reflecteix a la Figura 3-2. Les diferències bàsiques són dues. En primer lloc, les restriccions imposades sobre el disseny es tenen en compte durant tot el procés de generació. En segon lloc, el procés d'optimització està lligat a la generació del mòdul. Una eina que segueix aquest esquema es troba actualment en fase de desenvolupament.

La informació necessària pel procés de generació està constituïda per una descripció del circuit a nivell de transistors, l'especificació del conjunt de restriccions a complir i informació extreta del procés de síntesi lògica, com ara l'activitat de commutació de cada node del circuit, que pot ser útil en la construcció del mòdul final.

Amb independència de l'esquema de generació de mòduls utilitzat, hi ha una sèrie de tasques que s'han de realitzar, en un o altre ordre, considerant o no les restriccions de disseny o fins i tot integrant vàries tasques en una sola etapa. Aquestes tasques comprenen els processos de particionat del circuit, agrupament de transistors, posicionament, optimització del rendiment i connexionat, que a continuació seran descrits breument.

Particionament del circuit

Correspon al procés mitjançant el qual un circuit es divideix en subcircuits que seran tractats per separat. L'objectiu, és clar: reduir la complexitat dels processos posteriors treballant sobre els diferents subcircuits de forma seqüencial.

Agrupament de transistors

Aquest procés inclou les tasques de reordenació, aparellament i encadenament de transistors.

La *reordenació de transistors* consisteix en el canvi d'ordre dels transistors d'una etapa d'un circuit sense variar la seva funcionalitat. Aquesta reorganització dels transistors pot aconseguit millores en termes d'àrea [**MH90**], retard [**BCL95**] o consum [**MC96**].

Habitualment, encara que no sempre, els mòduls generats disposen els transistors P i N en dues fileres de difusió paral·leles [**UC81**]. Per tal de reduir el connexionat intern convé alinear verticalment transistors que comparteixin el mateix senyal de porta. La determinació dels transistors que aniran alineats correspon a l'*etapa d'aparellament.*

Un determinat nombre d'aquestes parelles connectades per adjacència formen una cadena. El *procés d'encadenament* intentarà formar cadenes el més llargues possible per tal reduir el nombre trencaments a la difusió i, per tant, l'àrea.

Els processos d'aparellament i encadenament de transistors estan molt relacionats i de fet realitzar els dos processos en una sola etapa millora els resultats, tal com s'ha mostrat en [**VRRPC96**].

Posicionament

Després de l'etapa anterior es disposa d'uns elements als que cal assignar una posició relativa. D'això s'encarrega el *procés de posicionament*. Com a unitats a posicionar s'escullen, típicament, les cel·les, les etapes o les cadenes de transistors.

Les alternatives que presenten els algoritmes per posicionament es poden dividir en dos grans grups: algoritmes deterministes i no deterministes. Els primers responen sempre amb el mateix resultat donat un circuit determinat. Això es deu a que

treballen amb un esquema que només permet decrèixer el valor de la funció de cost i, per tant, no poden sortir d'un mínim local. Els segons inclouen una variable aleatòria que els permet fugir dels mínims locals. Els algoritmes que obtenen millors resultats es troben entre aquests darrers. Entre aquests cal distingir la família dels algoritmes tipus *Simulated Annealing* [**KGV83**], i la família dels algoritmes genètics [**CP87**].

El que determina la solució cap a la que s'encamina l'algoritme és la funció de cost. Aquesta inclou les restriccions d'àrea, retard i consum del disseny, assignant a cadascuna d'elles un cert pes. En bona mesura el resultat de l'etapa de posicionament depèn de la qualitat de la funcions de cost. Dit d'una altra forma, la funció de cost està determinant el tipus de solució cap a la qual ens encaminem i, per tant, es tracta d'una part en la qual es requereix la major precisió possible.

Optimització

Dos són els processos d'optimització per excel·lència: el dimensionament de transistors i la inserció de *buffers*.

El *dimensionament dels transistors* és el procés pel qual es determinen les mides adequades dels transistors d'un circuit per tal de complir amb les restriccions de retard imposades al disseny. A més, cal que l'increment en consum i àrea sigui el mínim possible. L'estimació d'aquests paràmetres s'ha de dur a terme a través de simulacions. És en aquest punt on apareixen els problemes de modelització del circuit, simulació ràpida i eficient i extracció de camins crítics.

Tot i que fins ara l'etapa de dimensionament ha estat centrada en el retard introduït pels dispositius, cada cop té més incidència el retard degut a les línies de connexió. Per dissenys submicrònics, es fa cada cop més palesa la necessitat d'un dimensionament conjunt de transistors i amplades de línies de connexió [**MPP95**].

La inserció de *buffers* per tal de reduir el retard d'un circuit ha estat un tema abastament estudiat. Darrerament s'han presentat aproximacions al problema que minimitzen l'increment del consum [**TAA95**].

Connexionat

El *procés de connexionat* és l'etapa responsable de la connexió final dels nodes del circuit. Aquest procés es pot realitzar seguint mètodes concurrents, que consideren el connexionat de tots els nodes d'un circuit de forma simultània, o mètodes seqüencials, on els nodes es connecten un a un. Donada la dificultat de trobar algoritmes eficients pel primer cas, pràcticament en la totalitat dels casos s'utilitza un mètode seqüencial. D'aquest podem distingir dos grups: els algoritmes dirigits a la connexió de dos terminals, i els algoritmes que optimitzen la connexió d'un node multi-terminal. Els primers solen ser de concepció senzilla, i aconsegueixen bons resultats en un temps limitat. Els més utilitzats corresponen a la família d'algoritmes *Maze* [**Lee61**]. La seva extensió cap a l'ús d'un nombre *n* de capes de connexió és relativament fàcil. Al seu torn, els segons presenten un major grau d'optimització, però alhora requereixen temps d'execució molt més elevats. La gran majoria d'aquests algoritmes es basen en la utilització de arbres de tipus *Steiner*

[**HXKCH93**]. Els canvis necessaris per treballar sobre *n* capes de connexionat són força complexes.

Amb l'aparició de tecnologies que disposen de varies capes destinades al connexionat s'ha extès l'ús del connexionat *damunt les cel·les* [**BSH92**]. Consisteix bàsicament en reduir de forma dràstica la quantitat d'àrea destinada específicament al connexionat a base de permetre el connexionat per sobre dels transistors. Quan més elevat sigui el nombre de capes de metall menor serà l'àrea de connexionat necessària.

Síntesi Lògica orientada a generadors de mòduls

Al fer el lligam entre el procés de síntesi lògica i un generador de mòduls guiat per prestacions, els algoritmes clàssics de síntesi lògica es veuen modificats. La fase de reestructuració i optimització de la Xarxa Booleana no s'ha de veure gaire afectada, ja que no depèn directament de la tecnologia. Però el mapatge tecnològic canvia radicalment. En primer lloc, ja no es disposa d'un conjunt de cel·les de les quals cal escollir la millor implementació per a cada funció present en el circuit, sinó que *cal generar una implementació* per a cada funció en el circuit. En segon lloc, les funcions de cost que han de guiar el procés de mapatge tecnològic seran també diferents. Efectivament, en el cas del mapatge tecnològic sobre una biblioteca de cel·les estàndard, el cost de cada cel·la en termes d'àrea, retard i dissipació de potència és conegut durant el mapatge tecnològic, mentre que al considerar un generador de mòduls aquest cost no estarà definitivament fixat fins que el mòdul hagi estat generat. Ja veurem que aquest fet permet relaxar la dependència tecnològica durant el mapatge tecnològic. Les funcions de cost amb les quals es treballarà seran més senzilles i relativament independents de la tecnologia. Malgrat això, la bona correlació de les funcions utilitzades amb el resultat final del generador de mòduls permet assolir els compromisos exigits al circuit.

Conclusions

L'evolució tecnològia ha portat un increment creixent en la complexitat dels dissenys que es poden abordar. Les restriccions són cada vegada més estrictes i afecten a diferents objectius de disseny. Aquestes restriccions només són assolibles gràcies a una automatització creixent del procés de disseny d'un circuit integrat i al desenvolupament d'eines de disseny específiques que les tinguin en consideració durant tot el procés de disseny. Ara bé, per tal que aquestes eines puguin assolir compromisos cada cop més estrictes entre diferents objectius de disseny (àrea-retard-consum), cal proporcionar-los un ampli ventall d'implementacions possibles per cada funció, de tal manera que es pugui escollir entre un conjunt de compromisos diferents dels objectius de disseny. En la metodologia clàssica de dissenys parcialment a mida (*semi-custom*), això implica el desenvolupament de biblioteques amb un númerto cada vegada major d'especialitzacions per cada cel·la. En tot cas, però, tant el número de cel·les en la biblioteca com el número d'especialitzacions per cada cel·la serà forçosament limitat, donada la complexitat del desenvolupament i posterior manteniment d'una biblioteca de cel·les estàndard, i això malgrat les eines d'automatització existents per facilitar aquestes tasques.

Per solventar aquest problema, es proposa una metodologia nova, consistent en unir els processos de síntesi lògica i generació automàtica de mòduls guiada per prestacions. En aquest cas, no hi ha límit en quant al número de funcions o especialitzacions disponible. Així, la síntesi lògica disposa del ventall complet de funcions implementables en la tecnologia, pel que podrà assolir compromisos molt més restrictius sobre els objectius de disseny. La sortida del procés de síntesi lògica serà una descripció a nivell de transistors del circuit desitjat. Finalment, el generador de mòduls guiat per prestacions generarà les màscares del circuit, prenent en consideració la informació proporcionada per la síntesi lògica i respectant les restriccions originalment imposades al disseny.

La resta del treball exposat en aquesta tesi consisteix en l'anàlisi del procés de mapatge tecnològic orientat a generadors de mòduls guiats per prestacions, comparant-lo amb el procés clàssic de mapatge tecnològic sobre una biblioteca de cel·les estàndard. En el capítol 4 s'estudiarà la implementació d'una funció Boolena qualsevol. En el capítol 5 es mostrarà l'obtenció de totes les implementacions possibles d'un node de la Xarxa Booleana, incloent una part del seu ventall d'entrades transitives. En el capítol 6 es proposen algoritmes de cobertura del graf que representa el circuit i es comparen els resultats obtinguts usant la metodologia proposada i la clàssica de mapatge tecnològic sobre biblioteca de cel·les.

Capítol 4

Generació de Portes Complexes

En aquest capítol s'estudiarà el problema de generar una porta complexa, d'una sola etapa, que implementi una funció Booleana qualsevol representada per un BDD. S'estudiarà la possibilitat d'obtenir la implementació directament a partir de l'estructura del BDD. Es definirà el conjunt de funcions per a les quals aquesta implementació directa és possible a partir de l'estructura del BDD. També es definirà el conjunt de funcions per a les quals la implementació directa a partir de l'estructura del BDD resulta ser la implementació mínima. La resta de funcions seran implementades a partir d'una factorització màxima de la funció.

Introducció

Els BDDs constitueixen una representació compacta per a un ampli ventall de funcions Booleanes. Com s'ha comentat anteriorment, els BDDs presenten la interessant propietat de constituir una representació canònica per una funció Booleana, fixada una relació d'ordre entre les variables del suport de la funció. Aquesta propietat, juntament amb l'existència d'eficients algoritmes de manipulació de funcions expressades en BDDs, fa que aquests siguin la representació escollida per a una bona part de les eines de síntesi i verificació formal [**BRSW87**, **RPRVC95**, **RP93**] per tal de representar les funcions Booleanes manipulades internament.

Durant el mapatge tecnològic, les funcions Booleanes guardades en els nodes de la Xarxa Booleana han de ser implementades en la tecnologia destí. Quan el mapatge tecnològic es fa considerant una biblioteca de cel·les, el problema d'associar una funció amb una o més cel·les de la biblioteca es coneix amb el nom de problema d'*aparellament*. L'aparellament consisteix en obtenir el conjunt de cel·les de la biblioteca que poden implementar una determinada funció, considerant possibles permutacions i/o inversions de les entrades i/o de la sortida de la cel·la. El problema de l'aparellament ha estat ben estudiat i diversos algoritmes han estat proposats [**BL92**, **ZT94**, **WCA94**, **WH95**, **TZ96**]. En el cas que ens ocupa, però, la situació és

ben diferent. Quan es considera un mapatge tecnològic orientat a un generador de mòduls, no es disposa d'una biblioteca de cel·les de la qual se'n pugui extreure un conjunt de cel·les que implementin una funció determinada, sinó que cal generar explícitament una porta, a nivell de transistors, per aquesta funció. La porta generada ha de ser implementable en la tecnologia destí en una sola etapa, i ho serà sempre que no s'excedeixin uns certs límits tecnològics que suposarien una degradació excessiva en el senyal de sortida de la porta. Habitualment, aquest límit ve imposat pel número màxim de transistors en sèrie, tant de tipus P com de tipus N. A part de no excedir aquest límits tecnològics, és desitjable obtenir una porta amb el mínim número de transistors.

Donat que la nostra eina, *synthetic* (veure l'Apèndix D), utilitza BDDs com a representació interna de les funcions Booleanes, seria interessant disposar d'un algoritme que permetés obtenir la implementació a nivell de transistors directament a partir de l'estructura del BDD. Això ens permetria definir una funció de cost basada en l'estructura del BDD, de manera que les estimacions de les prestacions de la Xarxa Booleana fetes abans del mapatge tecnològic serien molt més senzilles, al no requerir la implementació prèvia dels nodes de la Xarxa.

Un algoritme per generar la implementació en portes DCVSL de funcions Booleanes representades en BDDs ha estat proposat en [**Cor95**]. Aquest treball té el seu origen en el món dels circuits assíncrons [**Lav94**] i es basa en el fet que les cel·les DCVSL disposen de les dues fases de cada variable d'entrada de la funció. Això permet fer una associació directe entre un arc del BDD i un transistor de tipus N etiquetat per la variable que controla el node on s'origina l'arc. La fase de la variable serà positiva si l'arc correspon a un fill *V* i negativa si l'arc correspon a un fill *F*. Aquest algoritme no és aplicable en el cas de considerar implementacions CMOS clàssiques, ja que no es disposa de les dues fases de cada variable d'entrada.

En [**RRAR95**] s'ha proposat un algoritme per generar portes CMOS complexes, d'una sola etapa, a partir d'un arbre de decisió binària no ordenat, que els autors anomenen TBDD, sobre el que s'han imposat una sèrie de restriccions. Un estudi detallat de l'algoritme proposat demostrarà que les restriccions imposades poden ser relaxades i que la tècnica proposada pot ser ampliada, per una certa classe de funcions, a BDDs ordenats.

Aquest capítol està organitzat de la següent manera. En primer lloc, es descriu breument l'algoritme proposat en [**RRAR95**]. Tot seguit s'estudiarà l'extensió de l'algoritme al cas de BDDs ordenats. Es demostrarà que tota funció implementable en una porta CMOS complexa pot ser implementada directament a partir de l'estructura del BDD que representa la funció. Malauradament, aquesta implementació no serà necessàriament òptima. Com a conseqüència d'això, s'analitzarà el conjunt de funcions que poden ser implementades òptimament a partir de l'estructura del BDD que les representa i s'estudiarà en quin percentatge aquestes funcions es troben en els circuits de prova proposats en [**Lsynth91**]. Finalment es presentaran els resultats de l'estudi efectuat i se n'extrauran les conclusions pertinents.

Del TBDD a l'Esquema Elèctric a Nivell de Transistor

L'algoritme proposat en [**RRAR95**] parteix de l'observació que una porta CMOS d'una etapa no és més que un conjunt *V* de variables v_i tal que cada variable v_i controla un parell de transistors N_i (de tipus N) i P_i (de tipus P). Aquests transistors es comporten bàsicament com interruptors. P_i és actiu i permet el pas d'informació únicament quan v_i val zero i, similarment, N_i només és actiu i permet el pas d'informació quan v_i val u. El valor a la sortida d'una porta CMOS ve definit per una combinació de valors de les entrades v_i de tal manera que si s'estableix una connexió elèctrica entre l'alimentació, V_{dd}, i la sortida, aquesta valdrà 1; si la connexió és entre el terra, V_{ss}, i la sortida, aquesta valdrà 0. Un BDD presenta una estructura similar, tal com es pot veure en la Figura 4-1. Un node del BDD està controlat per una variable v_i ; el fill *F* del node es pot assimilar a P_i i ell fill *V* a N_i. El valor de la funció *f,* representada pel BDD, també bé definit per la combinació de valors de les entrades v_i. Si les entrades estableixen un camí entre el terminal *1* i l'arrel, *f* valdrà 1; si el camí és entre el terminal *0* i l'arrel, *f* valdrà 0. La diferència bàsica, però, entre la representació en BDDs i la implementació en una porta CMOS està en el fet que en la porta CMOS, els transistors de tipus *P* i *N* es troben en plans diferents, mentre que en el BDD aquesta separació no existeix. L'algoritme proposat en [**RRAR95**] consisteix en restringir la topologia del BDD per tal de definir una subclasse que pugui representar cel·les complexes CMOS. Es consideraran únicament cel·les CMOS que compleixin les següents propietats bàsiques:

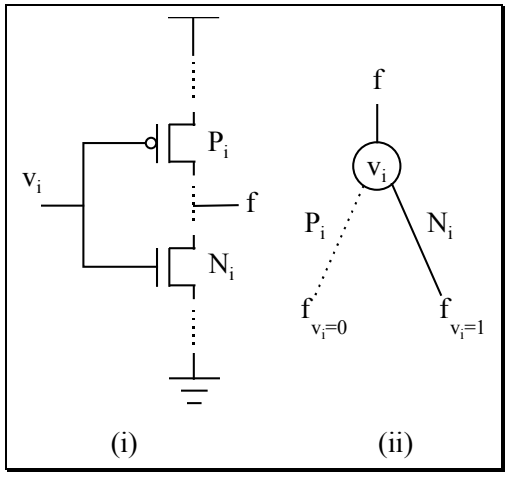

Figura 4-1 *Aspecte d'una entrada controlant dos transistors duals en una porta CMOS (i) i situació equivalent en un BDD (ii).*

CMOS i) Només estan connectats a l'alimentació, V_{dd}, transistors de tipus P_i.

CMOS ii) Només estan connectats al terra, V_{ss}, transistors de tipus N_i.

CMOS iii) Tots els transistors P_i es troben en un pla P, separat dels transistors N_i que es troben en un pla diferent N.

CMOS iv) Una porta que té més d'una entrada està formada sempre per una associació sèrie/paral·lela de transistors.

Si assimilem el terminal *1* dels BDDs a V_{dd}, i el terminal *0* a V_{ss} , les propietats *i)* i *ii)* es tradueixen en el fet que els BDDs implementables han de complir que només fills *F* poden arribar al terminal *1* i només fills *V* poden arribar al terminal *0*.

Partint de les propietats bàsiques de les portes CMOS, en [**RRAR95**] es defineix una estructura de dades anomenada TBDD, que correspon a un BDD no-ordenat que compleix les següents propietats:

TBDD i) Només arcs tipus P_i, fills *F*, estan connectats al terminal *1*.

TBDD ii) Només arcs N_i, fills V, estan connectats al terminal *0*.

TBDD iii) Tots els arcs que arriben a un node són del mateix tipus, sigui P_i o N_i.

TBDD iv) Existeix sempre un camí que passa per tots els nodes no terminals del TBDD.

TBDD v) Un BDD format per un únic node no terminal que compleixi les propietats *i)* i *ii)* anteriors és un TBDD. Els TBDDs més complexes són generats únicament a partir d'una associació sèrie/paral·lela de dos TBDDs.

L'estructura dels TBDDs proposada en [**RRAR95**] permet una fàcil associació entre arcs del TBDD i els transistors d'una porta CMOS equivalent. Abans, però, cal definir el concepte de precedència entre nodes.

Definició: Direm que un node n_1 precedeix un altre node n_2 en un BDD (o TBDD) si com a mínim existeix un camí que s'origina a n_1 i arriba a un node terminal passant per n_2.

La construcció del pla P d'una porta CMOS es pot fer aplicant les següents regles a un TBDD per tal d'associar transistors de tipus *P* als arcs P_i:

Regla i) El terminal *1* correspon a l'alimentació, V_{dd}.

Regla ii) L'arrel del TBDD correspon al node de sortida de la cel·la.

Regla iii) Els arcs P_i són transistors de tipus P tal que la porta està controlada per la variable v_i que etiqueta el node on s'origina l'arc. El drenador del transistor correspon al node on s'origina l'arc, i la font al destí de l'arc.

Regla iv) Si a un node hi arriben arcs de tipus N_i, l'arc que s'origina en un node que precedeix els nodes on s'originen la resta d'arcs N_i representa un curt-circuit; la resta són circuits oberts.

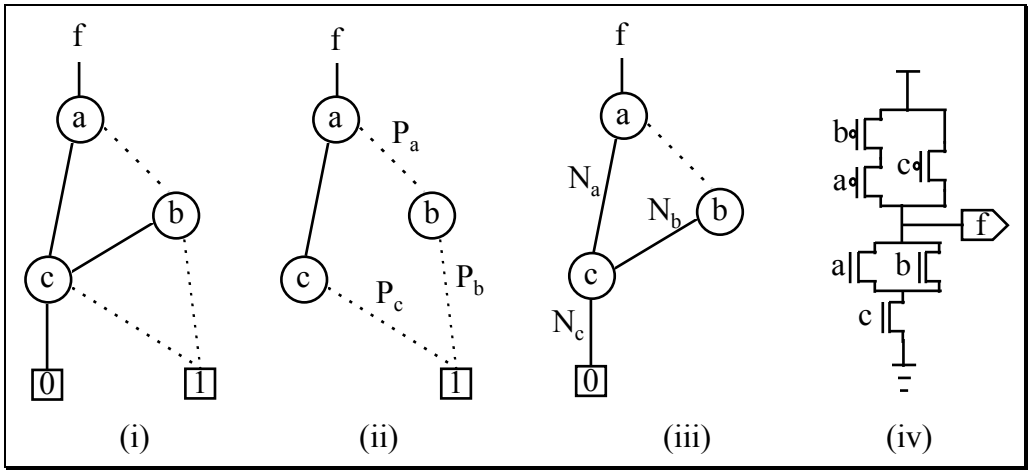

Figura 4-2 *Procés d'associació de transistors als arcs del TBDD de (i). En (ii) els transistors de tipus P s'associen als fill F dels nodes. En (iii) els transistors de tipus N s'associen als fills V dels nodes del TBDD. La porta resultant es pot veure en (iv).*

Regles similars permeten fer l'associació de transistors N als arcs N_i per tal de construir el pla N de la porta CMOS equivalent al TBDD. Per tant, podem constatar que s'assignarà un transistor a cada arc del TBDD: un transistor de tipus P a cada arc P_i i un de tipus N a cada arc N_i. En la Figura 4-2 es pot veure un exemple de generació d'una porta CMOS a partir del TBDD que representa la funció ($f = \overline{a}\overline{b} + \overline{c}$, $f = \overline{(a+b)c}$).

Del BDD a l'Esquema Elèctric a Nivell de Transistor

En l'apartat anterior s'ha revisat la tècnica presentada en [**RRAR95**] per la generació de portes complexes a partir d'un TBDD. Ara s'estudiarà la possible aplicació d'una estratègia semblant en el cas de considerar BDDs ordenats. En primer lloc, es revisaran les propietats associades als TBDDs i es considerarà la seva validesa pel cas dels BDDs, considerant que les regles d'associació de transistors als arcs del BDD són les mateixes que en el cas dels TBDDs.

Les propietats i) i ii) dels TBDDs són vàlides i necessàries per la generació d'una porta CMOS complexa a partir de l'estructura d'un BDD. Aquestes propietats són:

BDD i) Només arcs tipus P_i, fills *F*, estan connectats al terminal *1*.

BDD ii) Només arcs N_i, fills *V*, estan connectats al terminal *0*.

Demostració BDD i)*: La regla* i) *de generació de transistors en el pla P identifica el terminal* 1 *amb el node d'alimentació,* V_{dd}*, de la porta CMOS. Segons la regla* i) *de les portes CMOS, els únics transistors que poden estar connectats a l'alimentació són els de tipus P. La regla* iii) *de generació de transistors en el pla P associa un transistor de tipus P a cada arc* P_i*. Per tant, els únics arcs que poden arribar al terminal 1 són els* P_i **.-**

Una demostració semblant és fàcilment deduïble per a la propietat *BDD ii)*.

Les propietats *iii)* i *iv)* dels TBDDs, pel contrari, no són necessàries per obtenir una implementació correcta a partir de l'estructura del BDD. Observem la Figura 4-3 (i). Al node etiquetat amb la variable *d* hi arriben dos arcs de tipus diferent, un és P_i i l'altre N_i, fet que contradiu la propietat *iii)* dels TBDDs. A més a més, no existeix cap camí, amb origen a l'arrel, que pugui contenir a l'hora els nodes etiquetats amb les variables *b* i *c*, incomplint així la propietat *iv)* dels TBDDs. Considerem la duplicació del node que incompleix la propietat *iii)* dels TBDDs, **Figura 4-3** (ii), de moment sense qüestionar-nos el perquè. Si després apliquem les regles d'associació de transistors als arcs del BDD no reduït resultant, arribem a la implementació que es pot veure en la **Figura 4-3** (iii). La porta resultant és correcte, és a dir, implementa la funció representada pel BDD de la Figura 4-3 (i). Això no obstant, val la pena destacar que no es tracta d'una implementació on els plans P i N siguin estrictament duals. És a dir, el pla N no correspon al que s'obtindria al negar la funció implementada en el pla P i aplicar a continuació les Lleis de Morgan.

La propietat *iii)* dels TBDDs té el seu origen en el mètode de construcció dels TBDDs. Efectivament, la propietat *v)* dels TBDDs diu que aquests només es

construeixen per associació sèrie o paral·lela de dos TBDDs. Aquestes associacions impliquen una concatenació de les estructures dels dos TBDDs [**RRAR95**], de manera que els nodes terminals del primer TBDD són eliminats i els arcs que arribaven a ells són desviats cap a nodes del segon TBDD. En l'associació sèrie, els arcs que arribaven al terminal *1* es desvien cap al node arrel i els arcs que arribaven al terminal *0* es desvien al terminal *0* del segon TBDD. En l'associació paral·lela, els arcs que arribaven al terminal *0* es desvien cap al node arrel i els arcs que arribaven al terminal *1* es desvien al terminal *1* del segon TBDD. És evident que, usant aquest mètode de construcció, els únics nodes als quals s'afegeixen nous arcs són o bé el node arrel o bé un dels nodes terminals del segon TBDD. Al node arrel no hi arribava cap arc anteriorment, i com que ara només hi arribaran arcs provinents d'un dels terminals del primer TBDD, aquests arcs seran forçosament d'un mateix tipus, degut a les propietats *i)* i *ii)* dels TBDDs. En el cas dels nodes terminals, cal observar que els únics arcs que es desvien cap a ells provenen d'un terminal del mateix tipus, i per tant les propietats *i)* i *ii)* dels TBDDs garanteixen que aquests arcs sempre seran del mateix tipus dels que ja hi arribaven originalment. En el cas dels BDDs, no s'imposaran restriccions tant estrictes sobre la seva construcció com la propietat *v)* dels TBDDs, pel que una propietat equivalent a la *iii)* dels TBDDs no serà requerida. Un raonament similar es podria fer per la propietat *iv)* dels TBDDs.

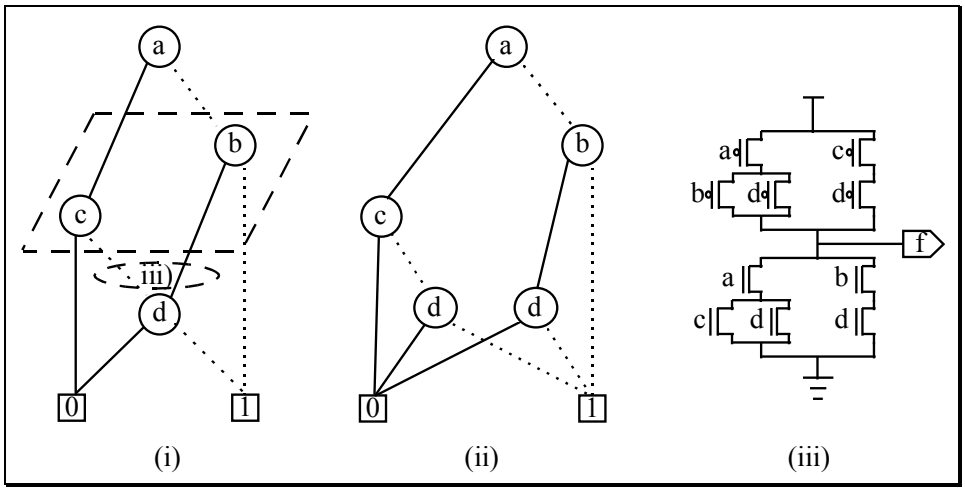

Figura 4-3 *En (i) es pot veure un BDD que no compleix les propietats* iii) *i* iv) *dels TBDDs. Malgrat això, si considerem la duplicació del node que no compleix la propietat* iii) *dels TBDDs, (ii), i després s'apliquen les regles d'associació de transistors als arcs del BDD, s'obté la implementació correcta, però no estrictament dual, mostrada en (iii).*

Ara bé, si a un node del BDD hi poden arribar arcs de diferents tipus, P_i i N_i, haurem de redefinir les regles d'associació de transistors als arcs del BDD, ja que les regles proposades pels TBDDs, de *i)* a *iv)*, no consideraven aquesta possibilitat, per estar en contradicció amb les propietats dels TBDDs. De fet, les regles *i)* a *iii)* continuen essent vàlides, i només caldrà modificar la regla *iv)*, que quedarà de la següent manera:

Regla iv) Si a un node hi arriba un o més arcs de tipus N_i, aleshores es procedirà de la següent manera. Es classificaran els arcs que arribin al node agrupant-los en base a la relació de precedència que mantinguin els diferents nodes on s'originen els arcs. Cada classe estarà formada per

tots aquells arcs els nodes origen dels quals mantinguin una relació de precedència entre ells. A efectes d'implementació, es considerarà un duplicat del node per cada classe definida. Així cada classe serà implementada separadament de la forma que s'indica a continuació. Els arcs de tipus P_i de la classe s'implementaran normalment, seguint les regles *i)* a *iii)* definides anteriorment. Pels arcs N_i, es nomenarà un representant, que serà aquell arc, el node origen del qual precedeixi la resta de nodes on s'originen els demés arcs N_i de la classe. Aquest representant s'implementarà com un curt-circuit; la resta com circuits oberts.

La present definició de la regla *iv)* explica perquè cal duplicar el node etiquetat amb la variable *d* en el BDD de la **Figura 4-3** (i). Al node hi arriben arcs de tipus N_i, pel que cal aplicar la regla *iv)*. Els nodes origen dels arcs que arriben al node etiquetat amb la variable *d* no compleixen cap relació de precedència entre ells, és a dir, no es troben en un mateix camí que vagi des de l'arrel al node terminal *1*. Per tant, es defineixen dues classes, una per cada arc. I segons la regla *iv)*, es considera una còpia del node per cada classe, pel que el node apareix duplicat en la **Figura 4-3** (ii). A partir d'aquí, l'aplicació de les regles d'associació de transistors als arcs del BDD ens porten a la implementació de la **Figura 4-3** (iii).

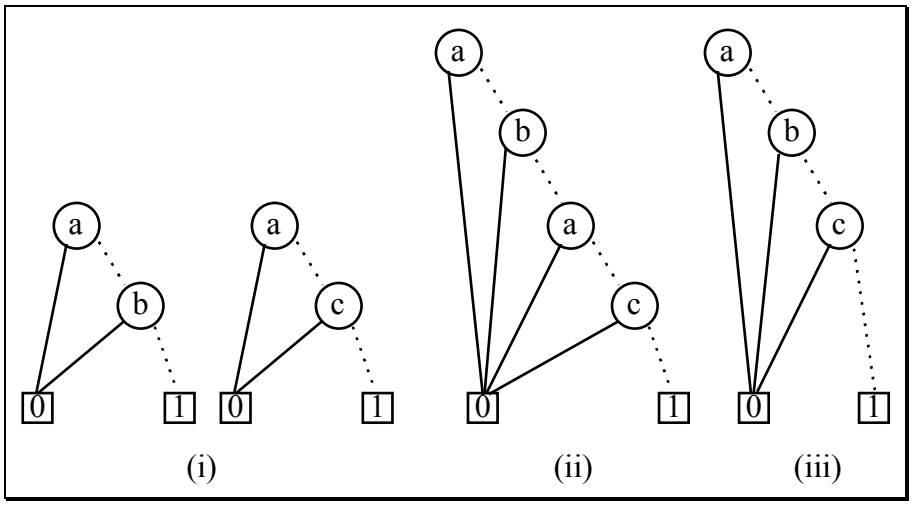

Figura 4-4 *Comparació entre la construcció en TBDDs o BDDs de la funció resultant de la multiplicació de les funcions mostrades en (i). En (ii) es pot veure el TBDD resultant. Es pot observar que la variable* a *apareix en dos nodes situats en nivells diferents. Aquesta situació no seria possible en un BDD (ordenat). Si l'operació es fa amb BDDs, el resultat és el que es mostra en (iii). Com es pot veure, la implementació resultant no conté redundàncies.*

Pel que fa a la propietat *v)* dels TBDDs, cal dir que aquesta no pot ser considerada en el cas dels BDDs. El motiu es troba en el fet que els BDDs són diagrames ordenats. Les variables en el suport del BDD han de respectar una relació d'ordre entre elles, pel que una mateixa variable no pot aparèixer dos cops en un mateix camí, ni tampoc en ordres relatius diferents entre si en camins diferents. Si la propietat *v)* dels TBDDs s'apliqués a dos BDDs amb alguna variable comuna en els seus suports, el resultat de l'operació ja no seria un BDD. Un exemple es pot veure en la Figura 4-4. Els suports

dels dos BDDs de la Figura 4-4 (i) comparteixen la variable *a*. L'aplicació de la propietat *v)* dels TBDDs (concatenació sèrie) sobre aquestes funcions produiria el resultat que es mostra en la Figura 4-4 (ii). L'existència de dos nodes controlats per la variable *a* en un mateix camí ens indica que el resultat no és un BDD ordenat. En realitat, el fet que en el cas dels BDDs no s'hagi d'aplicar la propietat *v)* dels TBDDs pot suposar fins i tot un avantatge. Efectivament, la Figura 4-4 (iii) ens mostra el resultat de la concatenació sèrie de les funcions de la Figura 4-4 (i), però aquest cop usant operacions de BDDs (en aquest cas, el producte de les funcions). El fet que el BDD sigui una representació canònica permet l'eliminació de nodes redundants. En aquest cas, ha suposat la desaparició d'un dels nodes controlats per la variable *a*. Si considerem la implementació a nivell de transistor a partir del TBDD de la Figura 4-4 (ii) i el comparem amb la implementació a partir del BDD de la Figura 4-4 (iii) podem observar com aquesta última requereix dos transistors de menys (recordem que s'associa un transistor a cada arc del TBDD o del BDD).

De fet, la propietat v) dels TBDDs proporciona el mètode de construcció d'aquests, i garanteix que els TBDDs construïts d'aquesta forma seran sempre implementables en portes CMOS aplicant les cinc regles esmentades. Si aquesta propietat no pot ser aplicada en el cas dels BDDs i es permet que aquests es construeixin en base a operacions generals entre ells, ens podem preguntar si de l'estructura d'un BDD qualsevol es podrà generar sempre una porta CMOS que implementi la funció representada pel BDD.

Podem respondre a la qüestió formulada en el paràgraf anterior constatant dues observacions simples. En primer lloc, cal destacar que no totes les funcions Booleanes poden ser implementades en una porta CMOS d'una etapa. Efectivament, només funcions de fase negativa, també anomenades funcions monòtones decreixents, poden ser realitzades en una porta CMOS d'una etapa. Aquest fet és evident, ja que els transistors de tipus P només són actius quan el senyal de porta es troba a nivell baix, és a dir, en fase negativa. En segon lloc, val a dir que un BDD pot representar qualsevol funció Booleana. D'aquí es dedueix que no totes les funcions representades en BDDs poden ser directament implementades en una porta CMOS d'una sola etapa. La qüestió, ara, consisteix en saber si totes funcions realitzables en una porta CMOS poden ser obtingudes directament a partir del BDD que les representa. La resposta la tenim en el següent Teorema:

Teorema. De tota funció de fase negativa se'n pot obtenir una implementació en una porta CMOS d'una sola etapa directament a partir de l'estructura d'un BDD que representi la funció.

Demostració (pla P de la porta CMOS): Consideri´s la generació del pla P de la porta CMOS a partir de l'aplicació de les regles d'associació de transistors P als arcs de tipus P_i *d'un BDD que representi la funció. Anem a demostrar que el pla P de la porta així generat correspon exactament a una cobertura de la funció representada en el BDD. Recordem que, en un BDD, la unió de tots els camins que porten al terminal* 1 *correspon a una cobertura de la funció representada en el BDD. Cada camí pot ser identificat amb un terme producte on apareixen únicament les variables que etiqueten els nodes que es troben en el camí considerat; la fase de la variable serà positiva si el camí, a partir del node considerat, segueix pel fill* V *o* N_i *i negativa si ho fa pel fill* F *o* P_i. *En el cas que ens ocupa, però, la funció*

representada pel BDD és de fase negativa. Per tant, tota variable que aparegui en un terme producte amb fase positiva és redundant i pot ser eliminada del producte. D'aquí es dedueix que la cobertura de la funció contindrà només aquelles variables associades a arcs P_i*, a l'igual que només s'associen transistors de tipus P als arcs* P_i*. És clar que els termes productes que s'originen en camins que només segueixen arcs de tipus* P_i *apareixeran idènticament en la porta resultant degut a l'aplicació de la regla* iii) *de generació de transistors. Per tant, només queda per provar que els termes producte que s'originen en camins que travessen arcs* N_i *apareixen també en el pla P de la porta.*

Considerem ara que arribem a un node v *del BDD a través d'un arc* N_i*. Sigui* p *el camí que porta de l'arrel al node* v*. Si a* v *no hi arriba cap altre arc, és clar que els termes producte parcials generats pels camins que porten de* v *al terminal* 1 *s'hauran de concatenar amb el producte parcial obtingut al seguir el camí* p*, de l'arrel a* v*. Observem que l'aplicació de la regla* iv) *d'associació de transistors als arcs* P_i *implementarà l'arc* N_i *que ens ha portat fins a* v *com un curt-circuit, que és justament l'equivalent a la concatenació dels productes parcials. Suposem ara que a* v *hi arriben diferents arcs. Podem classificar aquests arcs en classes tal com s'estableix en la regla* iv) *d'associació de transistors als arcs* P_i*. Els arcs que es troben en classes diferents provenen de nodes que no guarden una relació de precedència entre ells, de tal manera que és segur que mai no es poden trobar en un mateix camí que porti de l'arrel a* v*. Així, els termes producte generats en cada classe es poden considerar independentment de la mateixa manera que la regla* iv) *indica que les classes s'han d'implementar separadament. Considerem finalment una classe que conté diferents arcs que arriben al node* v*. En primer lloc, cal veure que els arcs de la classe seran del mateix tipus. Suposem que existís un arc de tipus* N_i *juntament amb un altre arc de tipus* P_i *a la mateixa classe. Dels nodes on s'originen els arcs, sigui* w *el que precedeix a l'altre i* x *la variable que l'etiqueta. La variable* x *apareixerà en fase positiva en algun terme producte que impliqui la funció i no serà redundant en el terme. Si això no fos així, el node* w *seria redundant en el BDD. La conclusió a la que hem arribat contradiu el fet que la funció representada en el BDD segui de fase negativa, per tant hem de concloure que tots els arcs d'una classe són de mateix tipus. Si tots els arcs són de tipus* P_i*, tindrem diferents camins parcials des de l'arrel al node* v*, que comparteixen després la subfunció que té l'arrel en* v*. Aquests camins parcials seran els mateixos que resultaran de l'aplicació de la regla* iv) *d'associació de transistors. Finalment, considerem el cas en el qual tots els arcs de la classe són de tipus* N_i*. Dels nodes on s'originen els arcs, sigui* w *el que precedeix als altres. Sigui* p' *el camí que porta de l'arrel a* w*. Sigui* p" *el camí que porta de* w *a un altre node on s'origina un arc. És evident que* p' *engloba* p'+p"*. Per tant, considerar* p' *és suficient i* p" *pot ser descartat. Això és el que indica la regla* iv) *de generació de transistors: l'arc corresponent a* w *s'implementarà com un curt-circuit i la resta com circuits oberts***.-**

Un raonament similar es podria deduir pel pla N de la porta CMOS. Per tant, queda clar que de tota funció de fase negativa se'n pot generar una porta CMOS que la implementi, directament a partir de l'estructura d'un BDD que representi la funció. Però, seguint el raonament de la demostració, és evident que aquesta implementació és equivalent a la cobertura que es pot extreure directament de l'estructura del BDD. Malauradament, aquesta cobertura no té perquè ser mínima i, el que és pitjor, depèn de l'ordre de les variables. Aquest problema serà analitzat en apartats posteriors.

Implementació de funcions no fase negatives

En l'apartat anterior s'ha analitzat el conjunt de funcions que poden ser implementades en una porta CMOS d'una etapa utilitzant l'associació directa de transistors als arcs d'un BDD que les representi. Aquest conjunt ha resultat ser el de les funcions de fase negativa, que són totes les funcions que poden ser implementades en una porta CMOS d'una etapa. En aquest apartat s'estudiarà la implementació de la resta de funcions Booleanes. Distingirem entre aquelles funcions que tenen alguna variable de fase positiva, però cap de fase binària, i les que contenen variables de fase binària. Les primeres es transformaran en funcions de fase negativa i podran ser implementades amb les tècniques proposades en l'apartat anterior, mentre que les segones seran implementades usant una representació en forma factoritzada de la funció.

Funcions amb variables de fase positiva

Considerarem com a funcions amb variables de fase positiva aquelles en les quals alguna variable és de fase positiva i no hi ha cap variable que sigui de fase binària. És a dir, la funció depèn o bé de la fase negativa, o bé de la fase positiva de les variables del suport, però mai de les dues fases d'una mateixa variable i, a més a més, hi ha com a mínim una variable de la qual la funció en depèn de la seva fase positiva. És evident que aquestes funcions són fàcilment transformables en funcions de fase negativa. Només cal fer un canvi de variable per les variables de fase positiva, de tal manera que la nova variable introduïda sigui equivalent a la negació de la variable substituïda. La funció resultant dependrà únicament de la fase negativa de les noves variables del seu suport. Serà, per tant, una funció de fase negativa a la qual se li podran aplicar les tècniques d'associació de transistors vistes en l'apartat anterior.

Sigui *f* una funció que conté com a mínim una variable de fase positiva i cap variable de fase binària. Sigui *X* el conjunt suport de *f*. El conjunt de variables de fase positiva de *f*, $X^{pos} \subseteq X$, es pot definir com:

$$X^{pos} = \left\{ x \middle| x \in X, f_{x=1} \supseteq f_{x=0} \right\} \tag{4-1}$$

Efectivament, si la funció depèn únicament de la fase positiva de la variable *x*, el cofactor de *f* respecte de *x* serà igual a tots els termes de la coberta de *f*, on la variable *x* ha estat eliminada dels termes producte on apareixia. Al seu torn, el cofactor de f respecte de $\bar{x}$ seran únicament els termes producte on no apareixia la variable *x* i, per tant, estarà sempre inclòs en el cofactor de *f* respecte *x*.

Un cop conegut el conjunt de variables de fase positiva, podem definir un conjunt de variables corresponents a la negació de les variables en X^{pos}, X^{inv}:

$$X^{inv} = \left\{ Y^{x_i} \middle| x_i \in X^{pos}, Y^{x_i} = \overline{x_i} \right\} \tag{4-2}$$

D'aquí podem obtenir una funció de fase negativa f^{neg}, equivalent a *f*, substituint en *f* les variables del conjunt X^{pos} per les del conjunt X^{inv}:

$$f^{neg} = f_{\left| x_i = Y^{x_i} \ \forall x_i \in X^{pos} \right.} \tag{4-3}$$

La funció resultant, f^{neg}, podrà ser implementada usant les tècniques d'associació directa de transistors als arcs d'un BDD que la representi, tal com les descrites en l'apartat anterior. Els elements del conjunt X^{inv} corresponen al número de variables que cal invertir a l'entrada de f^{neg}. La dimensió de X^{inv} no defineix exactament el número d'inversors necessaris per a la implementació de f^{neg}, ja que durant el mapatge tecnològic s'optimitza el número d'inversors introduïts escollint la fase amb la qual s'implementarà cada funció present en la Xarxa Booleana. Si, per exemple, una variable intermèdia de la Xarxa Booleana és requerida majoritàriament invertida en les funcions que es troben en el seu ventall de sortida, la funció implementada en el node que la genera serà la funció negada. D'aquesta manera s'estalviaran inversors.

Cal recordar que la implementació de la funció negada es pot extreure també directament de l'estructura d'un BDD que representi la funció. Efectivament, només cal intercanviar els plans P i N i la fase de les variables que controlen els transistors obtinguts de l'associació directa als arcs del BDD.

Funcions amb variables de fase binària

Fins ara s'ha estudiat la implementació de funcions de fase no binària. Tal com s'ha vist, aquestes poden ser implementades directament a partir de l'estructura d'un BDD que les representi, mitjançant l'associació de transistors als arcs del BDD. També s'ha definit un teorema que permet saber en quins casos aquesta implementació resulta ser la implementació òptima de la funció.

En aquest subapartat s'estudiarà la implementació de funcions de fase binària, que no poden ser implementades en base a les tècniques d'associació directa de transistors als arcs del BDD. Les tècniques que es descriuran podran ser utilitzades també per implementar aquelles funcions de fase no binària de les quals no es pot saber si la implementació directa a partir de l'estructura del BDD resultarà ser l'òptima.

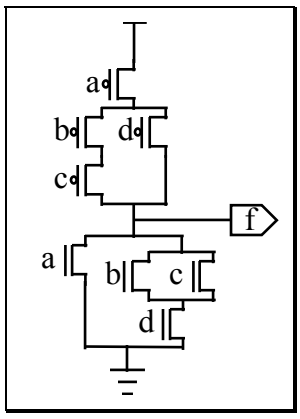

Figura 4-5 *Implementació de la forma factoritzada* $f = \overline{a}\left(\overline{b}\,\overline{c} + \overline{d}\right)$.

Si s'observa l'estructura del pla P d'una porta CMOS, es pot veure que aquest està format per la unió sèrie o paral·lela de grups de transistors que, al seu torn, estan formats per la unió sèrie o paral·lela d'altres grups de transistors i així successivament. De fet, aquesta estructura és isomòrfica a una representació de la funció en forma factoritzada. Si totes les variables apareixen en fase negativa en la forma factoritzada que representa la funció, la generació d'una porta CMOS d'una sola etapa que implementi la funció és directa. El pla P de la porta es construeix assignant a cada literal un transistor P controlat directament per la variable corresponent al literal i assimilant el producte en la forma factoritzada a una associació sèrie de transistors i la suma a una associació paral·lela de transistors.

El pla N de la porta es pot construir dualment assignant a cada literal un transistor N controlat directament per la variable corresponent al literal i assimilant la suma en la forma factoritzada a una associació sèrie de transistors i el producte a una associació paral·lela de transistors. Si la forma factoritzada conté literals en fase positiva, els transistors corresponents haurien d'anar controlats per la variable associada al literal, però negada.

Així, de qualsevol funció Booleana se'n pot obtenir la seva implementació CMOS en una porta d'una etapa a partir d'una representació en forma factoritzada. Cal destacar que la implementació de la funció complementària es pot obtenir també directament de la forma factoritzada. Només cal intercanviar els plans P i N, i complementar les variables que controlen els transistors. És evident que el cost d'aquesta implementació, quant a número de transistors, segueix essent el mateix que el de la implementació de la funció directa.

El procediment descrit per l'associació directa de transistors als literals que apareixen en una forma factoritzada identifica el problema de minimitzar la representació en forma factoritzada d'una funció a l'obtenció de la implementació mínima d'aquesta funció en una porta CMOS d'una etapa. A més a més, la factorització d'una funció Booleana és similar a la descomposició d'aquesta mateixa funció. La descomposició identifica i extreu subfuncions de la funció original, crea noves variables intermèdies per a elles i, finalment, expressa la funció original en base a les noves variables i a un subconjunt del suport original. La factorització, al seu torn, identifica també subfuncions comunes, però no les extreu, sinó que únicament les posa de relleu mitjançant l'ús de parèntesis en l'expressió que representa la funció. Aquesta semblança entre la descomposició i la factorització permet l'aplicació de tècniques semblants en els dos problemes.

Diverses tècniques de factorització o descomposició de funcions han estat proposades en la literatura [**BM82**, **Bra87**, **BHS90**, **HS92**, **Sch93**]. Per als nostres propòsits, les més interessants semblen ser les basades en l'operació de divisió lògica de funcions Booleanes. Cal dir que la utilització del terme divisió és incorrecte des del punt de vista matemàtic, ja que l'àlgebra de Boole no té inversa per a la multiplicació de funcions. No obstant això, degut a la semblança de l'operació a una

divisió clàssica en altres tipus d'àlgebres, el terme divisió està àmpliament tolerat i és emprat normalment en la literatura. Una breu descripció de l'operació de divisió i factorització segons els algoritmes proposats en [**BM82**, **Bra87**, **BHS90**] es pot veure en l'Apèndix C.

Un cop obtinguda una forma màximament factoritzada *F* per la funció *f*, cal generar la implementació en una porta CMOS d'una sola etapa de l'expressió *F*. Per això, només cal fer l'associació de transistors als literals de *F* tal com s'ha descrit anteriorment. No obstant això, cal recordar que l'aparició de literals en fase positiva obligarà a la introducció de variables intermèdies equivalents al complement de la variable associada al literal de forma similar a com es descriu en el subapartat *Funcions amb variables de fase positiva*. De la mateixa manera, caldrà introduir variables intermèdies associades a les variables de fase binària de *f*. En aquest cas, però, la nova variable no substitueix a l'anterior, sinó que les dues intervindran en la forma factoritzada, i també en la implementació final en una porta CMOS. La variable original s'associarà als literals que apareixen en fase negativa en la forma factoritzada, mentre que la nova variable s'associarà als literals que apareixien en fase positiva (i que ara apareixeran en la fase negativa de la nova variable).

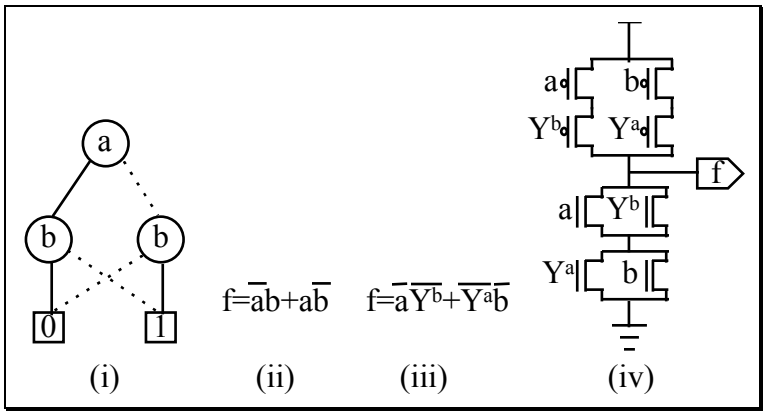

Figura 4-6 *BDD corresponent a una XOR (i). Expressió corresponent a la funció (ii), expressió implementable (iii) i implementació CMOS (iv).*

En la Figura 4-6 es pot veure la implementació d'una o-exclusiva (XOR) de dues variables. En la subfigura *(i)* es pot veure el BDD corresponent a la funció. L'associació directa de transistors als arcs del BDD produiria una implementació incorrecte, ja que la funció és de fase binària. La coberta de la funció, en *(ii)*, és no cub-factoritzable, és a dir, ja està màximament factoritzada. Les variables que apareixen en l'expressió són de fase binària. Cal introduir noves variables intermèdies equivalents al complement de les variables originals, seguint l'equació (4-2), i que substituiran els literals on les variables apareixen en fase positiva. El resultat d'aquesta substitució es pot veure en la subfigura *(iii)*. Ara, l'expressió està formada únicament per literals en fase negativa, i es pot fer l'associació directa de transistors als literals de l'expressió. Per construir el pla P de la funció, s'assimila el producte a una associació sèrie de transistors, i la suma a una associació paral·lela. Per pla N, l'associació és la inversa: el producte a una associació paral·lela i la suma a una associació sèrie de transistors. La implementació final en una porta CMOS d'una etapa es pot veure en la subfigura *(iv)*.

Del BDD a la implementació Òptima

En apartats anteriors s'ha estudiat com es pot obtenir, per a funcions de fase negativa, una implementació en una porta CMOS d'una sola etapa directament a partir de l'estructura d'un BDD que representi la funció. Per demostrar que això és possible per qualsevol funció de fase negativa, s'ha assimilat l'obtenció d'una coberta per la funció, extreta directament de l'estructura del BDD, amb el procés de generació d'una porta CMOS que implementi la funció. Malauradament, la coberta de la funció obtinguda de l'estructura del BDD dependrà, evidentment, d'aquesta estructura. I l'estructura del BDD depèn, al seu torn, de l'ordre de les variables de les quals depèn la funció. Llavors és fàcil concloure que la porta CMOS generada directament de l'estructura del BDD pel procediment descrit en l'apartat anterior no té perquè ser l'òptima.

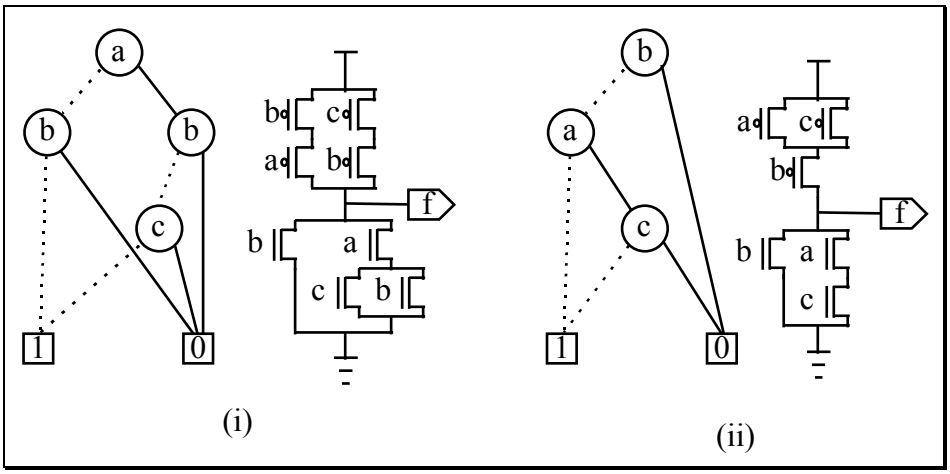

Figura 4-7 *En (i) es pot veure la implementació obtinguda a partir d'un BDD amb un ordenament de variables no òptim. En (ii) es pot veure la implementació obtinguda per la mateixa funció quan el BDD té un ordenament òptim de les variables. S'estalvien dos transistors.*

El procediment de generació d'una porta CMOS a partir de l'estructura del BDD assigna un transistor a cada arc del BDD. El número d'arcs en un BDD és igual a dues vegades el nombre de nodes no terminals. I el número de nodes no terminals depèn de l'ordenament de les variables presents en el BDD [**Bry86**]. En la Figura 4-7 es poden veure dues implementacions possibles per la funció $f = \overline{a}\overline{b} + \overline{b}\overline{c}$: en *(i)* el BDD que representa la funció s'ha construït amb les variables ordenades alfabèticament i la implementació resultant té 8 transistors; en *(ii)* el BDD s'ha construït amb un ordre diferent de les variables d'entrada i la implementació resultant necessita només 6 transistors. Un bon ordenament de les variables del BDD que representa la funció permet estalviar dos transistors.

El problema de l'obtenció de bons ordenaments de les variables per tal de reduir el número de nodes del BDD ha estat profusament estudiat, i diverses heurístiques han estat proposades en la literatura [**FMK91**, **BRKM91**, **MKR92**, **JPHS92**, **BBFS93**]. Un estudi detallat de la relació existent entre l'ordenament de les variables i l'estructura del BDD ha permès l'aparició de tècniques de manipulació incremental de l'ordenament de les variables, basades en intercanviar la posició de dues variables

adjacents en l'ordenament [**JCT91**]. Aquesta tècnica pot ser aplicada localment, cosa que la fa altament eficient, ja que la seva complexitat dependrà únicament del número de nodes afectats. Això ha portat al desenvolupament de tècniques de reordenament incremental de les variables del BDD [**FYBS93**, **Rud93**]. Actualment, els paquets de manipulació de BDDs solen incorporar tècniques de reordenament dinàmic de les variables, basades en intercanvis locals. Aquestes tècniques són aplicades automàticament quan es detecta un increment excessiu de l'espai ocupat pels BDDs, de tal manera que es busca un ordenament que minimitzi l'espai global ocupat per tots els BDDs definits i gestionats pel paquet.

Synthetic, com la majoria d'eines de síntesi i verificació de circuits, estableix un ordre global entre totes les variables que intervenen en la descripció del circuit. Això és necessari ja que, en cas contrari, abans de poder fer una operació qualsevol entre dos BDDs caldria establir un ordenament únic entre les variables del suport dels dos BDDs. En molts casos, el cost associat a l'establiment d'un ordre comú entre les variables dels dos BDDs seria molt més gran que el cost de realitzar l'operació desitjada entre els BDDs. Donada la quantitat d'operacions entre BDDs que intervenen en el procés de síntesi o verificació d'un circuit, el sobrecost que suposaria mantenir un ordenament diferent per cada BDD és inacceptable. Així, doncs, s'estableix un ordre únic, comú per tots els BDDs que intervenen. Gràcies a l'aplicació de tècniques de gestió dinàmica de l'ordenament, l'espai global ocupat pels BDDs es mantindrà, per a la majoria de circuits, en uns valors acceptables. Però és evident que l'ordenament òptim d'un BDD determinat pot no coincidir amb l'ordenament òptim global per tots els BDDs.

Del raonament anterior es desprèn que no és viable mantenir l'ordenació òptima per cada BDD que representi les funcions implementades en els nodes de la Xarxa Booleana. Per tant, seria convenient de saber si d'un BDD determinat es pot obtenir o no la millor implementació per a la funció representada pel BDD. Això equival a establir un criteri d'optimalitat per a la representació mitjançant un BDD d'una funció de fase negativa. Aquest criteri ens ve donat pel següent teorema:

Teorema. El número mínim de nodes no terminals necessaris per representar una funció en un BDD és igual a la dimensió del suport de la funció.

Demostració: Suposem que existís una variable x *en el suport de la funció per la qual no figurés cap node etiquetat per ella en un BDD que representés la funció. Considerem la coberta de la funció definida per la unió de tots els camins que porten al terminal* 1 *en el BDD. Aquesta coberta dependrà únicament de les variables que etiqueten els nodes del BDD. Per tant, no dependrà de* x*, fet que contradiu la suposició que* x *és una variable del suport de la funció. Per tant, el número mínim de nodes que tindrà un BDD que representi la funció serà igual a la dimensió del suport de la funció***.-**

El teorema ens diu que la representació mínima d'una funció en un BDD consta de tants nodes no terminals com variables té el suport de la funció. Un raonament equivalent ens permet establir quina serà la implementació òptima d'una funció per la qual el BDD que la representa té el número mínim de nodes no terminals.

Teorema. Si un BDD té un número de nodes no terminals igual a la dimensió del suport de la funció de fase negativa que representa, l'aplicació de les regles d'associació de transistors als arcs del BDD generarà una implementació òptima de la funció en una porta CMOS d'una sola etapa.

Demostració: Pel teorema anterior sabem que el BDD representa la funció amb el número mínim de nodes no terminals. Al ser la funció de fase negativa, la implementació directa a partir de l'estructura del BDD és possible. El número de transistors de tipus P serà igual al número de transistors N i igual al número de nodes no terminals del BDD, que és igual a la dimensió del suport de la funció. Considerem el pla P d'una porta CMOS que implementi la funció. Suposem que existís una variable x *en el suport de la funció per a la qual no figurés cap transistor P controlat per* x*. Considerem la coberta de la funció definida pel pla P de la porta. Aquesta coberta dependrà únicament de les variables que controlen els transistors del pla P. Per tant, no dependrà de* x*, fet que contradiu la suposició que* x *és una variable del suport de la funció. Un raonament semblant es pot deduir pel pla N. Per tant, podem concloure que no existeix cap implementació de la funció que tingui menys transistors que els generats per l'associació directa de transistors als arcs del BDD***.-**

Un cop definit el conjunt de BDDs dels quals sabem que se'n pot extreure la implementació òptima, en una porta CMOS d'una sola etapa, ens podem preguntar en quin percentatge aquests BDDs es troben presents en els circuits reals. Si el percentatge fos majoritari, es podria fer una primera avaluació del cost de la implementació del circuit basant-se només en els BDDs que representen les funcions del circuit. Efectivament, es podria calcular el número de transistors basant-se en el número de nodes dels BDDs. El número màxim de transistors P (N) en sèrie permet calcular el retard intrínsec de la porta i es pot calcular fent un recorregut del BDD, ja que és igual al número màxim d'arcs de tipus P_i (N_i) que conté un camí qualsevol des de l'arrel del BDD al node terminal *1 (0)*. Per la seva banda, l'activitat de commutació, relacionada amb la dissipació de potència, es pot estimar directament a partir de les funcions realitzades en els nodes de la Xarxa Booleana. En el capítol 6 es parlarà més extensament de l'avaluació de prestacions.

El percentatge de funcions que poden ser implementades òptimament a partir de l'estructura del BDD que les representa serà estudiat en l'apartat següent, juntament amb el percentatge total de funcions que poden ser implementades òptimament a partir d'una forma factoritzada.

Implementació Òptima de funcions Booleanes

La generalització del raonament aplicat per establir el mínim número de transistors necessaris per a implementar una funció de fase negativa ens permetrà d'establir el límit inferior al número de transistors necessaris per a implementar una funció Booleana qualsevol.

Teorema. El número mínim de transistors necessaris per implementar una funció Booleana en una porta CMOS d'una sola etapa és igual a dues vegades la dimensió del suport de la funció sumada al número de variables de fase binària.

Demostració (pel pla P):Per funcions de fase negativa, sabem que el número mínim de transistors necessaris per a la seva implementació és igual a dues vegades la dimensió del seu suport (demostrat en teoremes anteriors). Per a funcions amb alguna variable de fase positiva, però cap de fase binària, les transformacions descrites en el subapartat anterior ens permeten la transformació de la funció en una funció de fase negativa sense incrementar el seu suport i, per tant, es segueix complint el teorema. Només ens queda demostrar que el teorema és vàlid per a funcions de fase binària. Suposarem que totes les variables en el suport de la funció són de fase negativa o binària. Si existissin variables de fase positiva, s'aplicarien en primer lloc les transformacions del subapartat anterior per tal de transformar-les en variables de fase negativa. Considerem el pla P d'una porta CMOS que implementi la funció. Suposem que existís una variable x *en el suport de la funció per la qual no figurés cap transistor P controlat per* x*. Considerem la coberta de la funció definida pel pla P de la porta. Aquesta coberta dependrà únicament de la fase negativa de les variables que controlen els transistors del pla P. Per tant, no dependrà de* x*, fet que contradiu la suposició que* x *és una variable del suport de la funció. Suposem que existís una variable de fase binària* x *en el suport de la funció, per la qual no figurés cap transistor P controlat pel complement de* x*. Considerem la coberta de la funció definida pel pla P de la porta. Aquesta coberta dependrà únicament de la fase negativa de les variables que controlen els transistors del pla P. Si no hi ha cap transistor P controlat pel complement de* x*, no dependrà de la fase positiva de* x*, fet que contradiu la suposició que* x *és una variable de fase binària del suport de la funció. Per tant, el mínim número de transistors que figuren en el pla P és igual a la dimensió del suport de la funció sumat al número de variables de fase binària. Un raonament semblant es pot deduir pel pla N amb el que el número de transistors del pla P és multiplica per dos i el teorema queda demostrat***.-**

El límit teòric definit pel teorema ens permet saber si la implementació obtinguda per les funcions presents en un circuit és l'òptima. En la Taula 4-2 es pot veure una classificació de les funcions que es troben originalment en un conjunt de circuits de prova [**Lsynth91**]. Les columnes agrupades sota *Total* indiquen el total de funcions i transistors presents en els circuits. Les columnes agrupades sota *BDD* indiquen el total de funcions que han estat avaluades òptimament a partir de l'estructura del BDD que les representa. En aquesta columna es compten també aquelles funcions que tenen alguna variable de fase positiva. Finalment, les columnes agrupades sota *Òptim* indiquen el total de funcions que poden ser avaluades òptimament en el circuit. S'inclouen també les funcions no fase negatives que es poden implementar òptimament amb les tècniques de factorització proposades en l'apartat anterior. La subcolumna etiquetada amb *Func.* indica el número de funcions; la subcolumna etiquetada amb *Trans.* indica el número de transistors necessaris per implementar les funcions; la subcolumna *Inv. Tr.* indica el número de transistors que poden arribar a ser requerits per a la generació de la fase adequada en les entrades de les funcions corresponents.

	Total			BDD			Òptim		
Circuit	Func.	Trans.	Inv. Tr.	Func.	Trans.	Inv. Tr.	Func.	Trans.	Inv. Tr.
cm82a	6	52	18				4	32	16
cm151a	9	62	14	2	6		9	62	14
parity	15	120	60				15	120	60
cmb	14	120	38	10	68	24	13	104	34
cm163a	16	106	24	12	74	8	16	106	24
mux	6	104	24	1	4	4	1	4	4
cm162a	19	116	20	15	84	4	19	116	20
cm150a	16	124	32	1	4	2	16	124	32
cm85a	24	128	28	15	66	12	24	128	28
cu	23	172	40	16	100	20	22	154	36
pm1	31	170	54	28	46	50	31	170	54
pcler8	24	190	96	16	68	64	17	78	68
cc	33	198	46	18	78	30	33	198	46
count	47	316	110	31	188	46	47	136	110
my_adder	49	514	96	17	34		17	34	
C17	6	24		6	24		6	24	
C1355	546	2128	372	546	2128	372	546	2128	372
C1908	880	2994	738	880	2994	738	880	2994	738
C2670	1193	4150	2186	1193	4150	2186	1193	4150	2186
C3540	1669	5872	2820	1669	5872	2820	1669	5872	2820
C432	160	744	142	142	600	70	160	744	142
C499	202	1232	724	98	400	308	202	1232	724
C880	383	1458	602	383	1458	602	383	1458	602
TOTALS	4988	19636	7682	4716	16988	6758	4940	18710	7528
				94,55%	86,51%	87,97%	99,04%	95,28%	98,00%

Taula 4-2 *Nombre total (columnes* Total*) de funcions (*Func.*), transistors (*Trans.*) i transistors en inversors per generació de fase (*Inv. Tr.*) en els circuits de prova; nombre de funcions que poden ser implementades òptimament i directa de l'estructura del BDD (columnes* BDD*) i total de funcions que poden ser implementades òptimament (columnes* Òptim*).*

Els resultats de la Taula 4-2 mostren que prop del 95% de les funcions presents en els circuits poden ser avaluades òptimament a partir de l'estructura dels BDDs que les representen. Respecte al número de transistors, el percentatge es redueix lleugerament. Poc més del 86% dels transistors requerits pel circuit són deguts a les funcions que poden ser avaluades òptimament a partir de l'estructura del BDD. El percentatge arriba gairebé al 88% dels transistors que poden ser requerits per a la generació de fase per a les entrades de les funcions del circuit. En les columnes agrupades sota l'etiqueta *Òptim,* s'indica el percentatge de funcions que apareixen en els circuits de prova i que poden ser implementades òptimament, així com també els transistors que aquestes funcions representen. Els resultats mostren que el 99% de les funcions en els circuits de prova poden ser implementades òptimament. Quant al número de transistors, aquestes funcions representen el 95% del total de transistors, i arriben al 98% dels transistors que podrien ser necessaris per a la generació de la fase adequada en l'entrada de les funcions. Aquests últims inclouen també els inversors que poden ser necessaris per generar la fase complementada de les entrades de fase binària de les funcions presents en el circuit.

Els resultats que fan referència al percentatge de funcions que poden ser implementades òptimament a partir de l'estructura del BDD que les representa indiquen que es pot tenir una molt bona aproximació del cost inicial d'implementació d'un circuit usant únicament la informació estructural disponible en els BDDs que representen les funcions realitzades en els nodes de la Xarxa Booleana que representa el circuit. Així, l'avaluació dels cost d'implementació del circuit durant l'optimització de la Xarxa Booleana es pot basar en les estimacions estretes de l'estructura dels BDDs que representen les funcions del circuit. Aquest tipus d'estimacions poden ser usades també per les eines de síntesi de més alt nivell, com una acotació superior del cost d'implementació d'un circuit determinat, ja que, en general, l'optimització de la Xarxa Booleana i posterior mapatge tecnològic del circuit tendiran a reduir aquest cost.

Conclusions

A diferència de la metodologia clàssica basada en una biblioteca de cel·les estàndard, el mapatge tecnològic orientat a un generador de mòduls no disposa d'un conjunt de portes d'on escollir la millor implementació per un node determinat de la Xarxa Booleana, sinó que ha de generar la porta que necessita en cada moment. Per tant, en aquest capítol s'ha estudiat la implementació de funcions Booleanes en portes CMOS d'una sola etapa.

En primer lloc s'ha proposat la implementació a partir de l'estructura del BDD que representa la funció, per l'associació directa de transistors als arcs del BDD. S'ha demostrat que aquesta estratègia és possible per a totes les funcions implementables en una porta CMOS d'una etapa, és a dir, les funcions de fase negativa. També s'ha mostrat que una funció que contingui variables de fase positiva, però cap de fase binària, pot ser transformada en una funció de fase negativa i implementada directament a partir de l'estructura del BDD resultant.

Malauradament, l'estructura del BDD depèn de l'ordenament de les variables en el suport de la funció. Això implica que una implementació directa a partir de l'estructura del BDD no sempre portarà a la implementació òptima de la funció. Per a aquestes funcions i també per a les funcions de fase binària, s'ha proposat una implementació directa a partir d'una d'una representació màximament factoritzada de la funció. D'aquesta manera s'identifica el problema de la generació d'una forma factoritzada mínima amb el de la generació d'una implementació CMOS mínima, la qual cosa permet l'aplicació de les tècniques descrites en la literatura per la factorització de funcions al problema de generació de la implementació òptima en una porta CMOS d'una sola etapa.

D'altre banda, s'han definit els criteris que permeten conèixer el número mínim de transistors necessaris per implementar una funció Booleana. Aquest límit ens permet saber el conjunt de funcions que són implementades òptimament usant les tècniques descrites en aquest capítol. Un estudi efectuat sobre un conjunt de circuits de proves ha mostrat que la majoria de funcions presents en els circuits poden ser implementades òptimament. En concret el 99% de les funcions en els circuits, que suposen un 95% dels transistors totals necessaris per implementar les portes

associades als nodes de la Xarxa Booleana, i un 98% dels transistors que poden ser necessaris per inversors de generació de fase. Però segurament el resultat més interessant és el fet que prop del 95% de les funcions poden ser implementades òptimament a partir de l'associació directa de transistors als arcs dels BDDs que les representen. En número de transistors, suposen el 86% dels transistors que apareixen en les portes del circuit, i el 88% dels que poden ser necessaris en inversors per generació de fase a les entrades de funcions que no són de fase negativa.

Els resultats indiquen que, en el cas de considerar un mapatge tecnològic orientat a generadors de mòduls, es pot utilitzar la informació estructural disponible en els BDDs que representen les funcions dels nodes de la Xarxa Booleana per a obtenir una estimació acurada del cost del circuit en les primeres etapes de la síntesi lògica. En concret, durant la fase d'optimització lògica de la Xarxa Booleana, i també durant la síntesi d'alt nivell, es podrien utilitzar funcions de cost independents de tecnologia tal com el número de nodes del BDD (aproximació al número de transistors), número màxim de camins *F* o *V* des de l'arrel fins als terminals del BDD (equivalent al número màxim de transistors en sèrie, relacionats amb el retard intrínsec de la porta) i també l'activitat de commutació, que es pot calcular directament de la funció en el BDD (relacionada amb la dissipació de potència).

Proporcionar funcions de cost independents de tecnologia però estretament lligades als paràmetres de disseny durant les primeres etapes de la síntesi lògica, ha de permetre obtenir millors compromisos entre l'àrea ocupada pel circuit, el màxim retard i la potència dissipada. Efectivament, el millor lligam entre les funcions de cost utilitzades en l'optimització de la Xarxa Booleana i *la i*mplementació final del circuit permetrà obtenir un millor circuit a l'inici del mapatge tecnològic. Si a això hi unim el fet que l'increment de la granularitat del mapatge tecnològic orientat a un generador de mòduls, comparat al mapatge clàssic usant una biblioteca de cel·les possibilita un moviment més suau per l'espai de solucions, podem concloure que els compromisos entre els diferents paràmetres de disseny podran ser assolits amb més facilitat.

Capítol 5

El procés d'Agrupament

En aquest capítol es tracta el procés d'agrupament dels nodes de la Xarxa Booleana. Per cada node de la Xarxa Booleana es pretén crear una llista amb tots els possibles agrupaments de funcions que es troben en el seu ventall transitiu d'entrades i són implementables. En aquest capítol s'estudia la complexitat d'aquest procés i es proposen heurístiques per tal de reduir-la.

Introducció

El pas previ al mapatge tecnològic consisteix en l'optimització de la Xarxa Booleana, que crea una estructura que tendeix a compartir les subfuncions presents en els nodes de la xarxa per tal de reduir l'àrea global del circuit. A continuació, el mapatge tecnològic implementarà aquesta Xarxa Booleana en un circuit, minimitzant una certa funció de cost.

Si l'objectiu perseguit pel mapatge tecnològic no és únicament l'àrea, l'estructura de la Xarxa Booleana haurà de ser modificada lleugerament per tal d'adaptar-la als nous requeriments. Aquests canvis durant el mapatge tecnològic són possibles gràcies al procés d'agrupament, que genera totes les implementacions possibles de cada node de la Xarxa Booleana. L'elecció d'una o altre implementació durant la cobertura de la Xarxa Booleana possibilitarà la minimització de la funció de cost associada als objectius de disseny.

En aquest capítol s'estudiarà la problemàtica associada a la generació de tots els agrupaments implementables pels nodes de la Xarxa Booleana. En primer lloc, es definirà formalment el problema. A continuació s'estudiarà la seva complexitat i es proposarà una possible implementació. La complexitat del procés d'agrupament resultarà ser molt elevada, fet que motivarà el desenvolupament de tècniques encaminades a reduir-la. Aquestes tècniques seran aplicades a un conjunt de proves per tal de mesurar la seva eficàcia. Finalment es presentaran els resultats i se n'extrauran les conclusions pertinents.

Definició del problema

La generació de tots el possibles agrupaments implementables d'una Xarxa Booleana *G=(V,E)* consisteix en generar, per cada node $v \in V,\ v \notin X$, el conjunt de totes les funcions implementables del mateix. Aquestes funcions s'obtenen a partir de la funció del node, f^v, i del fet d'anar col·lapsant en ella totes les possibles combinacions dels agrupaments dels nodes intermedis de la Xarxa Booleana que es troben en el ventall d'entrada de *v*.

Un exemple d'aplicació d'aquest procés per a una Xarxa Booleana senzilla el podem veure en la Figura 5-1. Els nodes *a, b i c* són entrades primàries de la Xarxa Booleana i per a ells no es generen agrupaments (ja que no implementen cap subfunció i, per tant, no necessiten ser implementats). Els nodes v_1 i v_2 depenen únicament d'entrades primàries. La seva única implementació possible consisteix en la funció que s'implementa en el propi node, f^{v1} i f^{v2} respectivament, ja que en el seu ventall d'entrada no hi figura cap node intermedi. Finalment, el node de sortida *v* té quatre possibles implementacions. La primera d'elles correspon a la funció implementada en el node, f^v. Les restants s'obtenen a partir d'aquesta, substituint tota possible combinació de les variables intermèdies, Y^{v1}, Y^{v2}, per tota possible implementació d'aquestes.

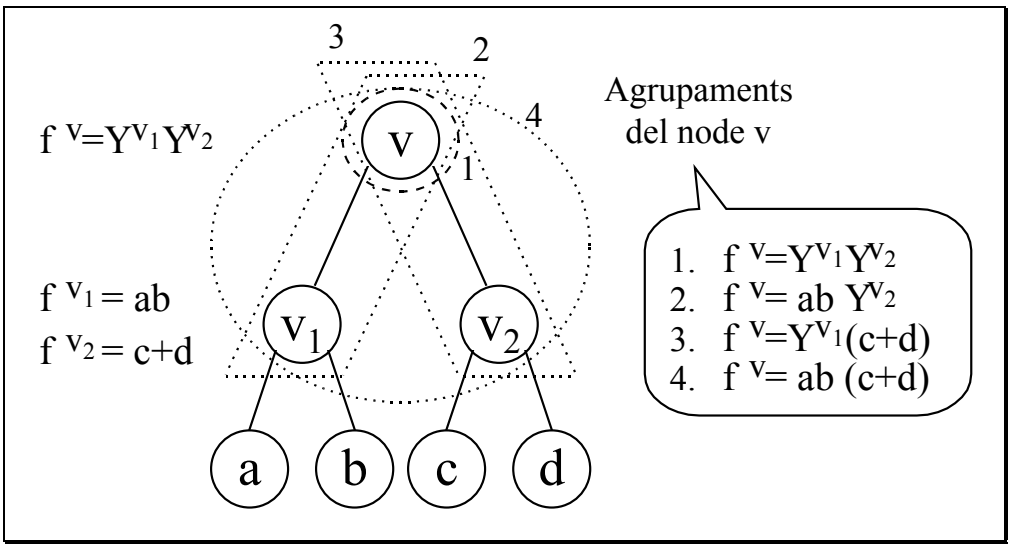

Figura 5-1 *Càlcul de tots els possibles agrupaments del node v.*

Abans de passar a formular matemàticament el problema de generació de tots els agrupaments implementables pels nodes d'una Xarxa Booleana *G=(V,E)*, caldrà definir una sèrie de conceptes. Sigui *X* el conjunt d'entrades primàries de *G*.

Definició: Anomenarem *grup* o *agrupament* d'un node $v \in V\text{-}X$ a tota funció *f'*, el ventall d'entrada de la qual conté únicament nodes presents en el ventall transitiu d'entrades de *v*, i que pot substituïr a f^v en el node *v* sense modificar en cap cas el valor de la variable de sortida del node, Y^v. De la definció resulta evident que la pròpia funció f^v constitueix un grup pel node *v*. En general, però, els agrupaments

del node *v* seran funcions *f'* que tenen un suport diferent de f^v. En tot cas, les variables del suport de *f'* han de pertànyer al ventall transitiu d'entrades de *v*.

Definició: Anomenarem *grup bàsic* d'un node $v \in V\text{-}X$ al grup constituït per la funció implementada en el node *v*, f^v.

Definició: Un grup serà *implementable* si la funció corresponent al grup pot ser implementada en la tecnologia destí.

En el cas que estem considerant, on el mapatge tecnològic s'orienta cap a un generador de mòduls, seran implementables totes aquelles funcions que no excedeixin el límit màxim de transistors en sèrie per la tecnologia. Aquesta restricció ve imposada per tal d'evitar l'excessiva degradació de les característiques del senyal de sortida de la porta.

Formalment, el problema de generació de tots els agrupaments implementables dels nodes d'una Xarxa Booleana $G = (V, E)$ consisteix en trobar els elements del conjunt *H(G)* següent:

$$H(G) = \left\{ h^v \,\middle|\, v \in \{V - X\} \right\} \tag{5-1}$$

on h^v és el conjunt d'agrupaments del node *v*, i està definit com:

$$h^v = \left\{ \bigcup_{0 \le i \le \left|V_e^v - X\right|} h_i^v \right\} \tag{5-2}$$

on h_i^v, $0 \le i < \left|V_e^v - X\right|$ representa el conjunt d'agrupaments generats degut a la substitució de la $i^{\text{éssima}}$ entrada del ventall d'entrades del node *v* (h_0^v representa el grup bàsic del node, on cap variable és substituïda). Els conjunts h_i^v es poden definir recursívament com:

$$\begin{aligned} h_0^v &= \left\{ f^v \right\} \\ h_i^v &= \left\{ \cup h = f_{\left|Y^{v_i} = g\right.} \,\middle|\, f \in \bigcup_{0 \le j < i} h_j^v,\ g \in h^{v_i},\ v_i \in \left\{ V_e^v - X \right\}, h \subset \text{implementables} \right\} \end{aligned} \tag{5-3}$$

Donada la definició anterior de h_i^v es podria pensar que el conjunt h^v d'agrupaments implementables del node *v* depèn de l'ordre en el qual es processen les variables del ventall d'entrades de *v*. Efectivament, l'ordre en el qual es processen els nodes del ventall d'entrades de *v* determinarà el contingut exacte del conjunts h_i^v. Sortosament, però, la substitució de *tots per tots* feta a l'equació (5-3), juntament amb el fet que la substitució de variables en una funció és una operació commutativa i associativa, garanteixen que la unió dels conjunts h_i^v donarà sempre el mateix resultat, h^v, independentment de l'ordre en el qual es considerin els nodes del ventall

d'entrades de *v* en l'equació (5-3). Cal observar també que el càlcul dels h_i^v en l'equació (5-3) només es podrà fer quan s'hagin calculat tots els possibles agrupaments pels nodes que es troben en el ventall d'entrades de *v*, és a dir, tots els conjunts $h^w, w \in V_e^v$.

Per clarificar el càlcul de tots els agrupaments implementables en una Xarxa Booleana, mostrarem a continuació el seu desenvolupament per a l'exemple de la Figura 5-1 a través de l'equació (5-1). Inicialement cal calcular els agrupaments h^v, h^{v_1}, h^{v_2} usant l'equació (5-2). Cal recordar que el càlcul h^v no es pot fer fins que s'hagin calculat els agrupaments dels nodes intermedis que es troben en el ventall d'entrades de *v*, és a dir,v_1 i v_2. Com que aquests dos nodes depenen únicament d'entrades primàries, la seva llista d'agrupaments contindrà només el seu grup bàsic. Veiem-ho:

$$h_0^{v_1} = \left\{ f^{v_1} = ab \right\} \qquad h_0^{v_2} = \left\{ f^{v_2} = c + d \right\}$$

$$h^{v_1} = \left\{ h_0^{v_1} = \{ab\} \right\} \qquad h^{v_2} = \left\{ h_0^{v_2} = \{c + d\} \right\}$$

A continuació passem a calcular h^v. Per fer-ho, utilitzarem les equacions (5-2) i (5-3) i les llistes d'agrupaments obtingudes anteriorment:

$$h_0^v = \left\{ f^v = Y^{v_1} Y^{v_2} \right\}$$

$$h_1^v = \left\{ f^v \circ f^{v_1} = ab\, Y^{v_2} \right\}$$

$$h_2^v = \left\{ f^v \circ f^{v_2} = Y^{v_1}(c + d), f^v \circ f^{v_1} \circ f^{v_2} = ab(c + d) \right\}$$

$$h^v = \left\{ Y^{v_1} Y^{v_2},\ ab\, Y^{v_2},\ Y^{v_1}(c + d),\ ab(c + d) \right\}$$

Noti's que si els nodes v_1 i v_2 es processen en un ordre diferent, els conjunts $h_i^v, i > 0$, canvien, però el conjunt final, h^v, no varia, només canvia l'ordre dels agrupaments. Per exemple, considerant una altre ordenació tenim:

$$h_1^v = \left\{ f^v \circ f^{v_2} = Y^{v_1}(c + d) \right\}$$

$$h_2^v = \left\{ f^v \circ f^{v_1} = ab\, Y^{v_2}, f^v \circ f^{v_2} \circ f^{v_1} = ab(c + d) \right\}$$

$$h^v = \left\{ Y^{v_1} Y^{v_2},\ Y^{v_1}(c + d),\ ab\, Y^{v_2},\ ab(c + d) \right\}$$

Complexitat del Procés d'Agrupament

En aquest apartat s'estudiarà la complexitat del procés de generació de tots els possibles agrupaments d'un node qualsevol d'una Xarxa Booleana *G=(V,E).* Per tal de dur a bon terme el nostre anàlisi hem de posar algun límit al ventall d'entrades que poden tenir els node de la Xarxa Booleana. Aquesta limitació vindrà imposada per la tecnologia o per la biblioteca de cel·les que s'estigui considerant. Suposarem, per tant, que el màxim ventall per cadascun dels nodes de la xarxa sigui de *k* entrades, tal com es pot veure en la Figura 5-2.

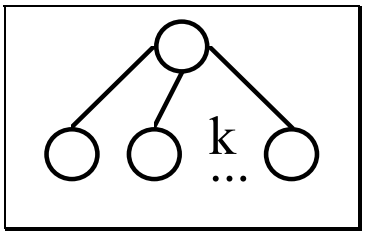

Figura 5-2 *Màxim ventall d'entrades (directes) per un node de la Xarxa Booleana.*

Ara be, el procés d'agrupament no té en compte únicament el ventall d'entrades directes, sinó que també considera el ventall d'entrades transitives del node. Per al desenvolupament del càlcul de la complexitat del procés de generació de tots els agrupaments implementables d'un node $v \in \{V - X\}$, classificarem el seu ventall d'entrades transitives per nivells topològics, tal com es pot veure en la Figura 5-3. Els nodes que es troben en el nivell zero serien aquells el suport dels quals està format únicament per entrades primàries de la Xarxa Booleana.

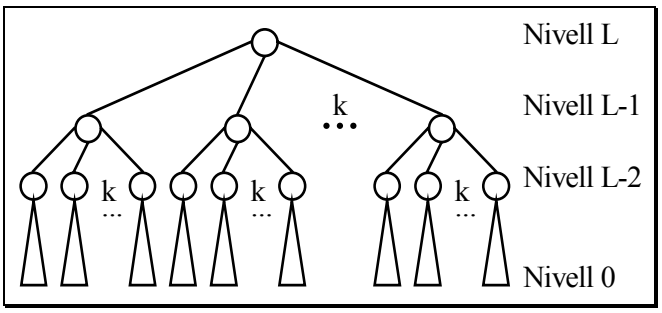

Figura 5-3 *Classificació per nivells del ventall d'entrades transitives d'un node de la Xarxa Booleana.*

És clar que la complexitat del procés de generació d'agrupaments dependrà bàsicament del número d'elements dels conjunts $h^w, w \in V_{et}^v$. Els elements de h^w es creen a través de l'aplicació recursiva de les equacions (5-2) i (5-3). Si *l* és el nivell topològic de *w,* anomenarem m_l al número d'elements màxim de h^w. Pretenen calcular m_l basant-nos en la definició recursiva de l'equació (5-3).

Els nodes *w* que es troben en el nivell zero només depenen d'entrades primàries de *G*. Per ells l'únic agrupament vàlid és el grup bàsic. Per tant, $\left|h^w\right| = m_0 = 1$. Podem calcular ara m_{l+1} basant-nos en m_l, conegut, i aplicant l'equació (5-3). Sigui *w* un node del nivell *l+1*. Calculem el número d'elements dels conjunts h_i^w:

$$\begin{aligned}
&\left|h_0^w\right| = 1 \\
&\left|h_1^w\right| = m_l \left|h_0^w\right| \\
&\left|h_2^w\right| = m_l\left(\left|h_0^w\right| + \left|h_1^w\right|\right) \\
&\vdots \\
&\left|h_i^w\right| = m_l \sum_{0 \le j < i} \left|h_j^w\right|
\end{aligned} \tag{5-4}$$

d'on podem calcular $m_{l+1} = \left|h^w\right|$:

$$m_{l+1} = \left|h^w\right| = 1 + m_l + m_l(m_l + 1) + m_l(m_l^2 + 2m_l + 1) + \cdots + m_l(m_l^{k-1} + \cdots + 1) \tag{5-5}$$

Tal com es pot veure en l'equació (5-5), el número d'elements de h^w és de l'ordre de:

$$m_{l+1} = O\left(m_l^k\right) \tag{5-6}$$

Això vol dir que per cada nivell que descendim en el ventall d'entrades transitives del node, *v*, que estem considerant el número d'agrupaments possibles creix exponencialment amb *k*, la dimensió màxima del suport d'un agrupament. En la Taula 5-1 es pot veure aquest creixement, en funció del nivell *L* on es trobi el node *v* i del número màxim de nodes en el ventall d'entrada, *k*. Els valors són reals, calculats a partir de l'expressió completa de l'equació (5-5).

	k=1	**k=2**	**k=3**	**k=4**
L=1	2	4	8	16
L=2	3	25	729	83521
L=3	4	676	3.890117×10^8	4.86635×10^{19}
L=4	5	458329	1.5133×10^{17}	5.60806×10^{78}

Taula 5-1 *Número de possibles agrupaments en funció del màxim ventall d'entrada, k, i del nivell del node, L.*

A la vista dels valors de la Taula 5-1 és evident el procés de generació de tots els possibles agrupaments per un node *v* de la Xarxa Booleana esdevé ràpidament intractable. D'altre banda, cal considerar que l'operació de composició d'agrupaments, que es realitza per a cada nou agrupament *h* en l'equació (5-3), té també una complexitat considerable:

$$O\left(f\big|_{x_i=g}\right) = O\left(|f|^2|g|\right) \qquad \text{[Bry86]} \quad (5\text{-}7)$$

on $|f|$ equival al número de nodes del BDD que representa la funció *f*.

El número màxim de nodes que poden ser necessaris per representar una funció mitjançant un BDD està acotat per:

$$|f| = 2^n/n \qquad (5\text{-}8)$$

on *n* és la dimensió del suport de *f*, és a dir, el número d'entrades de les quals depèn la funció.

Una simple observació de la Figura 5-3 ens fa evident que el número d'entrades creix proporcionalment a *k* per cada nou nivell de nodes que volem considerar. Sigui n_l la dimensió màxima del suport d'un agrupament qualsevol d'un node que es troba en el nivell *l*. És clar que n_l és igual a:

$$n_l = k^{l+1} \qquad (5\text{-}9)$$

	k=1	**k=2**	**k=3**	**k=4**
L=1	1	2	3	4
L=2	1	4	9	16
L=3	1	8	27	64
L=4	1	16	81	256

Taula 5-2 *Dimensió màxima del suport d'un agrupament del nivell L en funció del màxim ventall d'entrada, k.*

L'equació (5-9) ens permet calcular fàcilment la dimensió del suport màxim que podem tenir en un agrupament de nivell *L*, quan el ventall d'entrada màxim per un node és *k*. El resultat d'aquest còmput està tabulat en la Taula 5-2.

	$\|f\|$=O(2^n/n) [LL92]	O($\|f\|^3$)
n=1	2	8
n=5	7	343
n=10	103	1092727
n=15	2185	1.04316×10^{10}

Taula 5-3 *Ordre de magnitud de la dimensió del BDD que representa una funció de n entrades i complexitat de l'operació de composició de dues funcions de n variables.*

El creixement de la dimensió del suport dels agrupaments involucrats en la composició de funcions de l'equació (5-3) farà que les dimensions dels BDDs que els representen i la complexitat de l'operació de composició d'agrupaments creixin exponencialment, tal com es reflecteix en la Taula 5-3.

Sortosament, però, els únics agrupaments que interessa de calcular són aquells que després resultaran implementables en la tecnologia destí. Aquesta condició establirà un fort filtrat del número d'elements emmagatzemats en cada conjunt $h^w, w \in V_{et}^v$, reduint en gran mesura la complexitat de tot el procés.

Donada la complexitat associada a l'operació de composició de funcions de l'equació (5-3), el més interessant seria saber, *abans de generar un possible agrupament*, si aquest podrà ser finalment implementat en la tecnologia destí. De fet, si l'agrupament ha de resultar no implementable, no caldria generar-lo. S'estalviaria d'aquesta manera l'operació de composició de funcions i l'elevat cost associat a ella.

En aquest aspecte, la pròpia flexibilitat del procés de mapatge tecnològic orientat a generadors de mòduls es torna en contra seva. Efectivament, en el mapatge clàssic contra una biblioteca de cel·les estàndard, es coneixen efectivament les cel·les disponibles. No només això, sinó que les biblioteques de cel·les no solen contenir cel·les amb un número gaire elevat d'entrades. Aquest fet permet, d'una banda evitar en molts casos la composició de funcions, i de l'altre reduir la mida del conjunt de possibles agrupaments per cada node de la Xarxa Booleana. En el nostre cas, però, la limitació no està en el número d'entrades a una porta, sinó en el número màxim de transistors en sèrie que ens permet la tecnologia, per tal de no degradar les prestacions de la porta. Aquest fet permet un ventall d'entrada molt més elevat, reduint d'aquesta manera l'esmentada possibilitat de filtrat.

El número màxim d'entrades que pot tenir una porta CMOS sèrie paral·lel i dual implementada amb N_{max} transistors de tipus N en sèrie i P_{max} transistors de tipus P en sèrie és fàcilment deduïble a partir d'un anàlisi a nivell de branques de transistors. Efectivament, cada branca B_i de transistors P estarà associada a un grup de transistors N dual, que anomenarem G_i. Inicialment, suposarem que les branques B_i estan formades únicament per transistors en sèrie i, per tant, que els grups G_i estan formats únicament per transistors en paral·lel. El número màxim de transistors que podem tenir en una branca P correspon al màxim número de transistors P en sèrie que permet la tecnologia, P_{max}. Aquest número, multiplicat pel número de branques ens donarà el número màxim d'entrades que pot tenir una porta (una entrada per cada transistor P). Com que les branques B_i estan en paral·lel entre elles, els grups G_i estaran en sèrie entre ells. Aquest fet ens permet fixar el número màxim de branques B_i que serà igual al màxim número de transistors N en sèrie, N_{max}. D'aquí s'obté que el número màxim d'entrades per a una porta amb P_{max} i N_{max} transistors P i N en sèrie, respectivament, és igual a $N_{max}*P_{max}$, sempre i quan les branques B_i estiguin formades únicament per transistors en sèrie.

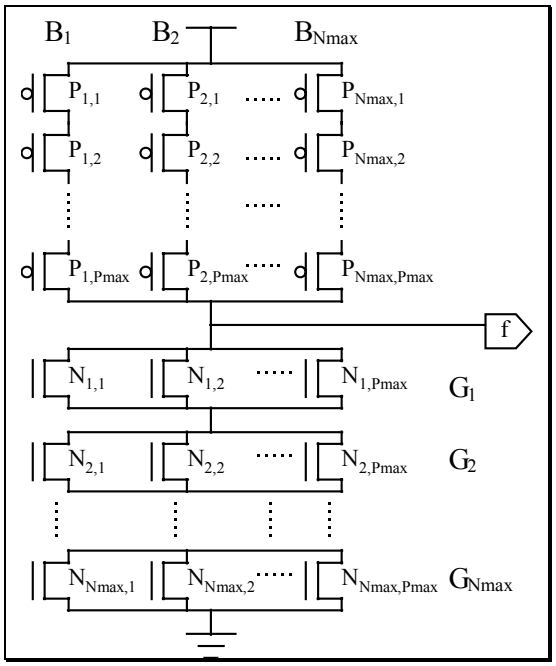

Figura 5-4 *Màxim número d'entrades per una porta dual amb P_{max} transistors P en sèrie i N_{max} transistors N en sèrie.*

Anem ara a estudiar el cas en el qual les branques B_i contenen transistors en paral·lel. És clar que aquests transistors en paral·lel tindran el seu equivalent dual en sèrie en el grup *Gi* corresponent. Suposem en primer lloc que introduïm un únic transistor en paral·lel amb el transistor $p_{j,r}, 1 \le r \le P_{max}$, en la branca B_j. El grup G_j tindrà dos transistors en sèrie. Donada la limitació en el número de transistors N en sèrie, això provocarà que el número total de grups *Gi* es redueixi en un (i per tant també ho farà el número total de branques B_i). Ara podem continuar introduint transistors en la branca B_j sense que es vegi reduït el número de branques total. No podem afegir cap més transistor en paral·lel amb el transistor $p_{j,r}$ ja que s'incrementaria el número de transistor en sèrie en el grup G_j i es tornaria a reduir el número de branques i grups. Tampoc podem introduir més d'un transistor en paral·lel per la resta dels transistors $s \neq r$ de la branca B_j. Per tant, el número màxim de transistors que podem tenir en la branca B_j és justament $2P_{max}$. Finalment, el número total d'entrades que pot tenir la porta és: $2P_{max} + (N_{\max} - 2) * P_{max} = P_{max} * N_{max}$.

	2 N sèrie	3 N sèrie	4 N sèrie	5 N sèrie	6 N sèrie	7 N sèrie
2 P sèrie	4	6	8	10	12	14
3 P sèrie	6	9	12	15	18	21
4 P sèrie	8	12	16	20	24	28

Taula 5-4 *Número màxim d'entrades en una cel·la en funció del número màxim de transistors en sèrie P i N permesos per la tecnologia.*

El número màxim d'entrades que pot tenir una porta amb P_{max} i N_{max} transistors P i N en sèrie, respectivament, el trobem tabulat a la Taula 5-4. Com es pot veure, i considerant que un ampli ventall de tecnologies fixaran valors de P_{max} per sobre de 3 i N_{max} per sobre de 4, el número màxim d'entrades a una porta pot ser relativament elevat, de l'ordre de la vintena d'entrades. La conseqüència és que la limitació de possibles agrupaments en l'equació (5-3) degut a la dimensió del suport és molt menys efectiva que en el cas tradicional de mapatge tecnològic usant una biblioteca de cel·les. La conclusió és que cal definir altres tipus d'estratègies per tal de limitar la complexitat del procés de generació de tots els possibles agrupaments pels nodes d'una Xarxa Booleana.

Implementació del procés d'agrupament

Una implementació natural del procés de generació dels possibles agrupaments pels nodes d'una Xarxa Booleana *G* és la solució recursiva proposada en l'Algoritme 5-1. Tots els possibles agrupaments dels nodes de la Xarxa Booleana es calculen recursivament a partir de les sortides primàries. Per a cada sortida primària, es calculen recursivament tots els agrupaments implementables del seu ventall transitiu d'entrades. Seguidament, es calculen els agrupaments implementables de la sortida en qüestió. El procediment *generaAgrupaments*, de l'Algoritme 5-1, s'encarrega de anar cridant recursivament a la funció *calculaAgrupaments*, que és la que efectivament calcula tots els agrupaments implementables pel node passat com a paràmetre.

```
funció generaAgrupaments(G)
  Per tot v∈sortides(G) fer:
    Per tot w∈V_e^v fer:
      calculaAgrupaments(w)
    calculaAgrupaments(v)
```

Algoritme 5-1 *Implementació del procés de generació de tots els agrupaments implementables per cadascun dels nodes d'una Xarxa Booleana G.*

La implementació de la funció *calculaAgrupaments* es pot veure en l'Algoritme 5-2. Es tracta d'una funció recursiva que, per poder generar tots els agrupaments

implementables pel node *v*, calcula i/o utilitza les llista d'agrupaments implementables dels nodes que pertanyen al ventall d'entrades de *v*. A continuació passarem a descriure el comportament de la funció.

```
funció calculaAgrupaments(v)
1)  Si no ∃ h^v fer:
2)      Crear h^v
        Sigui f^v la funció implementada en el node v
3)      Afegir f^v a h^v
4)      Per tot w ∈ V_e^v fer:
5)          calculaAgrupaments(w)
6)          h^act = h^v
            Per tot g ∈ h^w fer:
                Per tot h ∈ h^act fer:
7)                  Sigui f' la funció h on la variable w és subtituïda per la funció g
8)                  Si f' és implementable afegir f' a h^v
```

Algoritme 5-2 *Implementació del procés de generació de tots els agrupaments implementables per cadascun dels nodes d'una Xarxa Booleana G.*

En primer lloc, (línia 1), es comprova si ja s'ha creat la llista d'agrupaments del node actual. Aquesta comprovació es necessària, ja que un node pot tenir un ampli ventall de sortida i, per tant, pertànyer al mateix temps al ventall d'entrades de diversos nodes. Si no es fes aquesta comprovació, el càlcul dels agrupaments del node es repetiria per a cadascun dels nodes que pertanyen al seu ventall de sortida.

Un cop comprovat que no s'havia creat anteriorment la llista d'agrupaments pel node actual, es procedeix a la seva creació (línia 2). A continuació es genera el grup bàsic del node *v* i s'afegeix a la llista d'agrupaments (línia 3). El grup bàsic sempre ha de ser implementable. En cas contrari, podria no existir cap cobertura vàlida de la Xarxa Booleana. El procés d'optimització de la Xarxa Booleana haurà de garantir que es compleixi aquest requisit. En tot cas, els nodes que no són directament implementables poden ser (i han de ser) descompostos com a pas previ a la generació dels seus agrupaments.

El treball de major cost de la funció *calculaAgrupaments* es realitza en la iteració sobre el ventall d'entrades del node *v* (línia 4). Per cada element del ventall d'entrades, *w*, es calcula recursivament la seva llista d'agrupaments mitjançant una crida a la pròpia funció (línia 5). Seguidament es generen nous agrupaments pel node *v* a partir de la llista d'agrupaments del node *w*.

Val la pena destacar que tots els agrupaments que actualment es troben en la llista d'agrupaments de *v* depenen de la variable Y^w. Però els nous agrupaments que es generin a partir de la llista d'agrupaments de *w*, ja no dependran més de Y^w, sinó d'alguns elements del ventall d'entrades transitives de Y^w. Per això només podem generar nous grups a partir de la llista actual d'agrupaments de *v* i no podem considerar nous grups introduïts per *w* fins al següent pas de la iteració sobre el

ventall d'entrades (línia 4). Per tant, generem una còpia de la llista d'agrupaments actuals de *v* abans de començar a generar nous grups a partir de *w* (línia 6).

Tot seguit, la variable Y^w present en cadascun dels agrupaments actuals de *v* és substituïda successivament per cada element de la llista d'agrupaments de *w*, generant d'aquesta manera un nou grup pel node *v*. Si aquest grup és implementable, serà afegit a la llista d'agrupaments de *v* (línia 8). Tal com s'ha comentat anteriorment, els nous agrupaments introduïts seran utilitzats en el següent pas de la iteració sobre el ventall d'entrades (línia 4). D'aquesta manera, en cada pas de la iteració sobre el ventall d'entrades, la llista d'agrupaments de *v* es pot veure incrementada en un número de grups igual al producte de les longituds de les llistes d'agrupaments (actuals) de *v* i *w*.

La complexitat inherent al procés de generació d'agrupaments pels nodes de la Xarxa Booleana es pot reduir difícilment. Efectivament, la reducció o eliminació de possibles agrupaments implementables es tradueix en una reducció efectiva de l'espai de solucions disponibles per a la cobertura de la Xarxa Booleana, i això pot fer perdre la solució òptima. Per tant, posar limitacions sobre el número d'agrupaments màxim que es pot mantenir per a cada node, o sobre el nivell màxim de les entrades transitives que cal considerar, sembla poc adequat. En el primer cas, la pèrdua de solucions pot ser potencialment molt gran, depenent del número d'agrupaments descartats en cada node; en el segon cas, es descartarien algunes de les millors solucions en quant a possible optimització del retard màxim de la Xarxa Booleana, ja que els agrupaments complexes redueixen el número d'etapes.

L'única via efectiva per tal de reduir raonablement la complexitat del procés de generació dels agrupaments dels nodes de la Xarxa Booleana, sense afectar a l'espai de solucions, sembla ser la d'evitar la generació d'aquells agrupaments que després resultaran no implementables (línies 7 i 8 de l'Algoritme 5-2). Aquesta tècnica evita la generació de grups que tot seguit seran descartats al no ser implementables, segons l'equació (5-3).

Filtrat d'Agrupaments no Implementables

Tant en la discussió sobre la implementació com en l'estudi de la complexitat del procés de generació de tots els possibles agrupaments pels nodes d'una Xarxa Booleana s'ha comentat que l'única via efectiva de reduir la complexitat de tot el procés passa per evitar la generació d'aquells agrupaments que després resultaran no implementables en la tecnologia destí. S'estalvia així el costós procés de la composició de funcions en l'equació (5-3) sense reduir l'espai de solucions. En aquest sentit, el present apartat mostra l'estudi de diverses heurístiques de filtrat d'agrupaments. Aquestes tècniques es poden classificar en dos grans grups, segons sigui la idea sobre la qual es fonamenten:

- **Estimació de la complexitat de l'agrupament resultant**. Tracten de predir quina serà la complexitat de la funció resultant de la composició de funcions, usant únicament la informació disponible sobre els grups a composar.

- **Definició jeràrquica dels agrupaments**. Estableixen una jerarquia entre els diferents agrupaments que pot tenir cada node de la Xarxa Booleana i tracten de predir la complexitat dels nivells inferiors de la jerarquia en base als nivells superiors.

A continuació es passarà a descriure en detall cadascuna de les tècniques proposades i es comentaran els resultats obtinguts.

Estimació de la Complexitat de l'Agrupament Resultant

Les tècniques de filtrat d'agrupaments no implementables que es basen en fer una estimació de la complexitat de l'agrupament resultant, provenen de la translació i extensió de la tècnica usada durant el procés de generació d'agrupaments del mapatge tecnològic orientat a biblioteques de cel·les estàndard. Aquesta tècnica consisteix en calcular la dimensió del suport de l'agrupament resultant i comparar-lo amb el número màxim d'entrades disponible en la biblioteca de cel·les destí.

En el cas del mapatge tecnològic orientat a un generador de mòduls, ja hem vist que les consideracions sobre la dimensió del suport no són suficients degut al fet que el màxim suport permès per la tecnologia pot ser relativament elevat i hi hauria pocs agrupaments que podrien ser filtrats usant la dimensió del suport com a referència. De fet, en el cas del mapatge tecnològic orientat a un generador de mòduls, la condició que impedeix que una funció sigui implementable en la tecnologia destí és que s'excedeixi el número màxim de transistors en sèrie permesos per la tecnologia. Per tant, es tractarà de predir, a partir de la informació disponible dels agrupaments originals, si l'agrupament resultant de la composició de funcions de l'equació (5-3) excedirà o no aquest límit tecnològic. Tres diferents heurístiques han estat estudiades i implementades:

- **MaxSerial**. Consisteix en filtrar els agrupaments quan el número màxim de transistors en sèrie s'ha assolit en l'agrupament original. S'espera que la composició dels agrupaments introduirà nous transistors en sèrie que faran que el grup resultant sigui no implementable.

- **AddSerial**. Filtra els agrupaments en els quals la suma dels transistors en sèrie en les funcions originals, menys 1, excedeix el màxim permès per la tecnologia. Es resta un pel fet que, en la funció original, desapareix el transistor associat a la variable intermèdia que és substituïda. Com en el cas anterior, s'espera que la composició de les funcions originals provocarà que s'excedeixi el límit tecnològic en l'agrupament resultant.

- **IncSupport**. Aquesta heurística calcula el suport del nou agrupament. D'aquesta manera es pot calcular l'increment en la dimensió del suport entre la funció sobre la qual es fa la composició i l'agrupament resultant. Aquest increment es suma al número de transistors en sèrie existents en la funció original, on es fa la composició. Es filtraran aquells agrupaments on el resultat d'aquesta suma excedeixi el límit màxim de transistors en sèrie permesos per la tecnologia. Aquest filtre és més costós que els dos anteriors, ja que implica el càlcul del suport de l'agrupament resultant. Aquest càlcul suposa el recorregut dels BDDs associats a les funcions originals, amb una

complexitat de l'ordre de la dimensió dels BDDs. Però malgrat la seva major complexitat, és de preveure que aquesta heurística donarà millors resultats que les dues anteriors, ja que aquestes últimes no tenen en compte que els suports de les funcions a composar poden no ser disjunts. Si els suports no són disjunts, hi ha la possibilitat que part dels transistors en sèrie existents en la funció original siguin compartits en la funció resultant de la composició i que, per tant, no s'incrementi el número de transistors en sèrie.

Les estratègies que proven de predir la complexitat de l'agrupament resultant de la composició de funcions en l'equació (5-3) suposen que la composició introduirà nous transistors en sèrie, sigui al bloc P o al bloc N, i tracten d'avaluar quan s'excediran els límits imposats per la tecnologia. Malauradament, la composició de funcions no sempre implica un augment del nombre *màxim* de transistors en sèrie. Hi ha dos motius pels quals això pot no passar. En primer lloc, tal com s'ha comentat anteriorment, si els suports de les funcions a composar no són disjunts, pot ser que la complexitat de la funció resultant de la composició no augmenti; fins i tot podria passar que disminuís. En segon lloc, l'increment de transistors en sèrie es pot produir en una branca diferent de la que tenia el *màxim* número de transistors en sèrie abans de la composició. D'aquesta manera, pot ser que el nombre màxim de transistors en sèrie no es vegi alterat per la composició.

L'observació anterior ens porta a concloure que aquest tipus d'heurístiques filtraran erròniament un conjunt, més o menys important, d'agrupaments implementables. Per tant, no hauran de ser aplicades en aquells casos on interessi per damunt de tot arribar a la solució òptima en el procés de cobertura de la Xarxa Booleana. Per altre banda, és de suposar que la majoria d'agrupaments no implementables seran eliminats considerant aquest tipus d'heurístiques, cosa que accelerarà en gran mesura el procés de generació d'agrupaments.

Definició Jeràrquica dels Agrupaments

En el present treball s'ha estudiat també un altre tipus de filtrat dels agrupaments no implementables en l'equació (5-3), que es basa en la definició jeràrquica dels agrupaments. Efectivament, si s'observa la definició dels conjunts h_i^v, $0 \le i < \left| V_e^v - X \right|$ en l'equació (5-3) resulta evident que la generació dels agrupaments es fa de forma jeràrquica. Abans de generar els agrupaments per un node *v*, s'han d'haver generat tots els agrupaments dels nodes que es troben en el seu ventall transitiu d'entrada, ja que aquests són necessaris pel càlcul dels grups h_i^v. A més a més, l'operació de composició de funcions, en general, suposarà un increment de la complexitat de la funció resultant respecte de les funcions originals. Així, l'agrupament bàsic d'un node *w* serà, normalment, l'agrupament més senzill que tindrà el node, ja que la resta d'agrupaments hauran estat generats composant aquest agrupament bàsic amb agrupaments dels nodes del ventall d'entrada de *w*.

Aquesta observació ens porta a establir una estratègia de filtrat pels conjunts h_i^v, $0 < i < \left|V_e^v - X\right|$ que consisteix en el següent: si l'agrupament resultant de la composició d'un agrupament *f* del node *v* amb l'agrupament bàsic *g* del node $w \in \left\{V_e^v - X\right\}$ és no implementable, es descartaran la resta d'agrupaments del conjunt h^w per aquest agrupament *f* de *v* (veure l'equació (5-3)). Aquesta estratègia l'anomenem **List**.

En l'heurística *List*, la jerarquia dels agrupaments ha estat usada únicament per distingir entre el grup bàsic i la resta de grups d'un node. Però un estudi més profund del mètode de construcció dels conjunts h^v ens permetrà d'observar que aquests conjunts s'estructuren en una jerarquia molt més detallada. Efectivament, podem establir una classificació dels agrupaments d'un node basant-nos en la classificació de l'agrupament a partir del qual han estat generats. Anomenarem $C_{i,n}^v$ a la n*éssima* classe d'agrupaments definits en h_i^v. Anomenarem $\left|C_i^v\right|$ al número de classes d'agrupaments definits en el conjunt h_i^v. Aleshores podem establir la següent classificació dels agrupaments del node *v*:

$$C_{0,0}^v = h_0^v$$

$$C_{i,n}^v = \left\{ \cup h = f\Big|_{Y^{v_i}=g} \;\middle|\; f \in \bigcup_{0 \le j < i} h_j^v,\ g \in C_{l,m}^{v_i},\ v_i \in \left\{V_e^v - X\right\},\ n = m + \sum_{p=0}^{l-1} \left|C_p^{v_i}\right|, h \subset \text{implementables} \right\} \quad (5\text{-}10)$$

Definides les classes $C_{i,n}^v$, es proposa l'heurísta següent: si l'agrupament resultant de la composició d'un agrupament *f* del node *v* amb el primer agrupament *g* de la classe $C_{i,n}^w$ del node $w \in \left\{V_e^v - X\right\}$ és no implementable, es descartaran la resta d'agrupaments de la classe. És a dir, la resta d'elements de la classe no seran composats amb l'agrupament *f* de *v*. A més a més, si *i=n=0*, es descartaran tots els agrupaments dels conjunt h^w. Cap d'ells serà composat amb l'agrupament *f* del node *v*. Aquesta estratègia l'anomenem **Hierarchy**. És clar que l'estratègia *List* proposada anteriorment no és més que un cas particular de l'heurística *Hierarchy*, on només es consideren les classes $C_{0,0}^w$ pel node *w*.

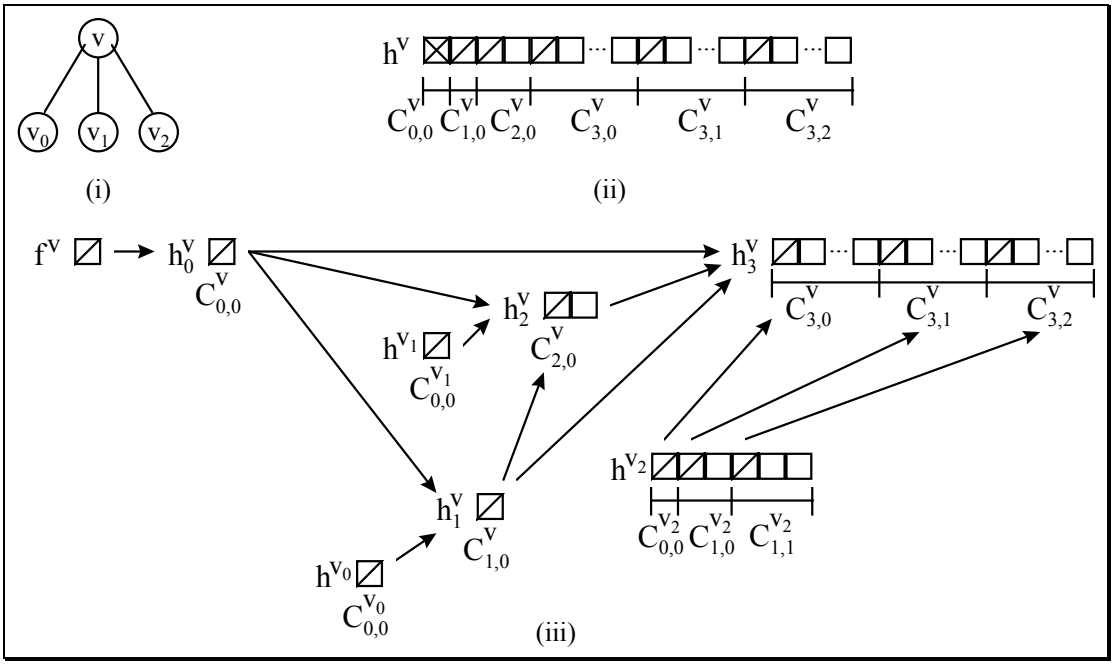

Figura 5-5 *Procés de generació i classificació d'agrupaments pel node* v. *Cada quadrat representa un agrupament. En (i) es veu el node* v *i el seu ventall d'entrades. En (ii) es veu el conjunt d'agrupaments pel node. En (iii) es mostra el procés de generació d'aquests agrupaments i com queden classificats durant el procés de generació. En* h^v*, els quadrats no barrats corresponen als agrupaments susceptibles de ser descartats per l'heurística* Hierarchy*, degut a una classe* $C_{i,n}$*,* $i \neq 0$*.*

Podem veure un exemple de com es generen les classes $C^v_{i,n}$ en la Figura 5-5. En *(ii)* es pot veure el conjunt d'agrupaments generats per *v*, i també quins d'aquests agrupaments són susceptibles de ser descartats per l'heurística *Hierarchy*. Els nodes no barrats en *(ii)* són susceptibles de ser descartats usant una classe $C^v_{i,n}$ amb $i \neq 0$. Tots els nodes de h^v, excepte el primer, assenyalat amb una creu, podríen ser descartats usant la classe $C^v_{0,0}$. Aquest darrer filtrat seria equivalent al comportament de l'heurística *List*.

Les heurístiques de filtrat d'agrupaments no implementables basades en una classificació segons l'origen dels agrupaments seran, en general, molt més conservadores que les heurístiques que proven de predir la complexitat de l'agrupament resultant, ja que les primeres únicament refusen agrupaments després de saber que la composició amb un agrupament bàsic resulta no implementable.

Resultats

En aquest apartat es compararan les diverses estratègies proposades pel filtrat d'agrupaments no implementables durant el procés de generació de tots els possibles agrupaments pels nodes d'una Xarxa Booleana. A ben segur els paràmetres més interessants a l'hora de fer la comparació entre les diverses tècniques seran el número

d'agrupaments descartats correctament i el número d'agrupaments implementables descartats erròniament. El primer ens mesura l'eficàcia, en quant a nivell de filtrat d'agrupaments no implementables, mentre que el segon ens mesura la possibilitat de descartar agrupaments necessaris per arribar a la solució òptima durant el procés de cobertura de la Xarxa Booleana.

Per tal de fer una comparativa significativa, s'ha escollit un conjunt de circuits de talla mitjana del conjunt de circuits de prova proposats en [**Lsynth91**]. La generació d'agrupaments s'ha fet sobre els circuits originals, sense cap mena d'optimització prèvia. En primer lloc, s'ha procedit a aplicar les heurístiques de tal manera que únicament s'anéssin classificant i comptabilitzant els agrupaments implementables o no, però sense permetre que el filtrat fos efectiu. D'aquesta manera s'han pogut comptabilitzar realment els agrupaments que intervenen en el procés de generació d'agrupaments, cosa que ha permès determinar l'eficàcia de les heurístiques. S'han diferenciat quatre categoria de grups:

- *Acceptats*. Correspon als agrupaments que serien correctament acceptats, és a dir, que passarien el filtre i resultarien ser implementables. Aquesta categoria ens mesura d'alguna manera les possibilitats d'obtenció de compromisos entre àrea, retard i dissipació de potència durant la posterior etapa de cobertura de la Xarxa Booleana.

- *No filtrats*. Serien els agrupaments que passarien el filtre, però que després resultarien no implementables. Correspon al conjunt d'agrupaments que podrien ser filtrats per l'heurística i que no ho són.

- *Filtrats*. Són els agrupaments que serien filtrats correctament, és a dir, que no passen el filtre proposat i que després resulten ser no implementables. Aquest grup ens mesura l'eficàcia, quant a filtrat, de la tècnica proposada.

- *Perduts*. Correspon als agrupaments que són filtrats erròniament per l'heurística proposada, i que resultarien ser implementables de passar el filtre. Aquest grup ens mesura la possibilitat de pèrdua de solucions òptimes en la cobertura de la Xarxa Booleana pel fet d'aplicar l'heurística.

A continuació es mostraran les taules de resultats obtingudes per cadascuna de les heurístiques proposades. A més a més de la classificació d'agrupaments esmentada anteriorment, les taules contenen el número total d'agrupaments (no bàsics) que conté cada circuit, etiquetat *Total*, així com quants són implementables, *Impl.*, i quants no, *No Im.*. D'aquesta manera es poden calcular els percentatges d'agrupaments existents en cada categoria. La darrera fila de la taula conté els resultat global de l'aplicació de l'heurística sobre el conjunt de circuits de prova. Els resultats més interessants es troben en les columnes d'agrupaments *filtrats*, que ens mesura l'eficàcia del filtre, i en la columna d'agrupaments *perduts*, que ens diu quants agrupaments implementables són erròniament descartats. Les caselles buides indiquen que cap agrupament pertany a la categoria corresponent.

D'entrada, val la pena observar que més del 56% del agrupaments que intervenen en el procés de generació d'agrupaments resulten ser no implementables i, per tant, susceptibles de ser filtrats per les heurístiques proposades. Aquest fet és important, ja que ens indica fins a quin punt es pot optimitzar el procés de generació d'agrupaments mitjançant la utilització de tècniques de filtrat.

Circuit	Total	No Im.	Impl.	Acceptats		No filtrats		Filtrats		Perduts	
cm82a	4	2	2	2	100,00%	2	100,00%				
cm151a	53	17	36	25	69,44%	1	5,88%	16	94,12%	11	30,56%
parity	39	25	14	14	100,00%	25	100,00%				
cmb	14	10	4	4	100,00%	7	70,00%	3	30,00%		
cm163a	27	19	8	8	100,00%	16	84,21%	3	15,79%		
mux	5	4	1	1	100,00%	4	100,00%				
cm162a	118	87	31	27	87,10%	68	78,16%	19	21,84%	4	12,90%
cm150a	158	99	59	49	83,05%	3	3,03%	96	96,97%	10	16,95%
cm85a	1072	607	465	242	52,04%	153	25,21%	454	74,79%	223	47,96%
cu	18	8	10	10	100,00%	6	75,00%	2	25,00%		
pm1	27	3	24	22	91,67%	1	33,33%	2	66,67%	2	8,33%
pcler8	373	242	131	64	48,85%	26	10,74%	216	89,26%	67	51,15%
cc	29	7	22	22	100,00%	7	100,00%				
count	159	55	104	99	95,19%	25	45,45%	30	54,55%	5	4,81%
my_adder	47	30	17	17	100,00%	30	100,00%				
TOTALS	2143	1215	928	606	65,30%	374	30,78%	841	**69,22%**	322	**34,70%**
		56,70%	43,30%								

Taula 5-5 *Resultats per l'heurística* MaxSerial.

Els resultats de l'aplicació de la primera heurística proposada, *MaxSerial*, es poden veure en la Taula 5-5. Tal com es pot observar, aquesta estratègia pot descartar prop del 70% dels agrupaments no implementables. Malauradament, també filtra, i per tant descarta erròniament, prop d'un 35% d'agrupaments que serien implementables.

Circuit	Total	No Im.	Impl.	Acceptats		No filtrats		Filtrats		Perduts	
cm82a	4	2	2	2	100,00%	2	100,00%				
cm151a	53	17	36	32	88,89%			17	100,00%	4	11,11%
parity	39	25	14	14	100,00%	19	76,00%	6	24,00%		
cmb	14	10	4	4	100,00%	6	60,00%	4	40,00%		
cm163a	27	19	8	8	100,00%	16	84,21%	3	15,79%		
mux	5	4	1	1	100,00%			4	100,00%		
cm162a	118	87	31	25	80,65%	60	68,97%	27	31,03%	6	19,35%
cm150a	158	99	59	47	79,66%			99	100,00%	12	20,34%
cm85a	1072	607	465	165	35,48%	81	13,34%	526	86,66%	300	64,52%
cu	18	8	10	10	100,00%	3	37,50%	5	62,50%		
pm1	27	3	24	22	91,67%	3	100,00%			2	8,33%
pcler8	373	242	131	75	57,25%	12	4,96%	230	95,04%	56	42,75%
cc	29	7	22	15	68,18%	6	85,71%	1	14,29%	7	31,82%
count	159	55	104	104	100,00%	25	45,45%	30	54,55%		
my_adder	47	30	17	17	100,00%	15	50,00%	15	50,00%		
TOTALS	2143	1215	928	541	58,30%	248	20,41%	967	**79,59%**	387	**41,70%**
		56,70%	43,30%								

Taula 5-6 *Resultats de l'heurística* AddSerial.

En la Taula 5-6 es poden veure els resultats de l'aplicació de l'heurística *AddSerial*. Prop del 80% dels agrupaments no implementables són correctament filtrats. Malauradament, l'heurística elimina també prop del 42% dels agrupaments correctes.

Circuit	Total	No Im.	Impl.	Acceptats		No filtrats		Filtrats		Perduts	
cm82a	4	2	2	2	100,00%	2	100,00%				
cm151a	53	17	36	16	44,44%			17	100,00%	20	55,56%
parity	39	25	14	14	100,00%	25	100,00%				
cmb	14	10	4	4	100,00%	5	50,00%	5	50,00%		
cm163a	27	19	8	8	100,00%	18	94,74%	1	5,26%		
mux	5	4	1					4	100,00%	1	100,00%
cm162a	118	87	31	31	100,00%	54	62,07%	33	37,93%		
cm150a	158	99	59	30	50,85%			99	100,00%	29	49,15%
cm85a	1072	607	465	215	46,24%	74	12,19%	533	87,81%	250	53,76%
cu	18	8	10	9	90,00%	5	62,50%	3	37,50%	1	10,00%
pm1	27	3	24	23	95,83%	1	33,33%	2	66,67%	1	4,17%
pcler8	373	242	131	111	84,73%	27	11,16%	215	88,84%	20	15,27%
cc	29	7	22	22	100,00%			7	100,00%		
count	159	55	104	104	100,00%	33	60,00%	22	40,00%		
my_adder	47	30	17	17	100,00%	30	100,00%				
TOTALS	2143	1215	928	606	65,30%	274	22,55%	941	**77,45%**	322	**34,70%**
		56,70%	43,30%								

Taula 5-7 *Resultats de l'heurística* IncSupport.

En la Taula 5-7 es poden veure els resultats de l'aplicació de l'heurística *IncSupport*. Més del 77% dels agrupaments no implementables són correctament filtrats. Malauradament, l'heurística elimina també prop del 35% dels agrupaments correctes.

A continuació es mostraran les taules de resultats corresponents a les heurístiques basades en la jerarquia dels agrupaments (*List, Hierarchy*). En aquestes taules s'han eliminat les columnes *Total, No Im. i Impl*., que serien idèntiques a les mostrades en les taules anteriors. A canvi contenen una columna més, etiquetada *No Pos*., que indica quants agrupaments dels que són susceptibles de ser filtrats són, però, no-filtrables per l'heurística. Cal recordar que aquestes heurístiques es basen en el càlcul de la composició d'agrupaments bàsics i en comprovar si el resultat és o no implementable. El filtrat consisteix en considerar o no els agrupaments derivats d'aquests agrupaments bàsics. Per tant, els agrupaments bàsics no podran ser filtrats en cap cas per aquest tipus d'heurística. En la segona fila de *Totals* hi ha el total d'agrupaments filtrats i no filtrats considerant el conjunt d'agrupaments bàsics que l'heurística no pot filtrar.

Circuit	Acceptats		No filtrats		Filtrats		Perduts		No Pos.	
cm82a	2	100,00%	2	100,00%					2	100,00%
cm151a	36	100,00%	14	82,35%	3	17,65%			1	5,88%
parity	14	100,00%	19	76,00%	6	24,00%			7	28,00%
cmb	4	100,00%	8	80,00%	2	20,00%			8	80,00%
cm163a	8	100,00%	14	73,68%	5	26,32%			13	68,42%
mux	1	100,00%	4	100,00%					4	100,00%
cm162a	31	100,00%	48	55,17%	39	44,83%			10	11,49%
cm150a	59	100,00%	80	80,81%	19	19,19%			3	3,03%
cm85a	447	96,13%	455	74,96%	152	25,04%	18	3,87%	25	4,12%
cu	10	100,00%	7	87,50%	1	12,50%			6	75,00%
pm1	24	100,00%	3	100,00%					1	33,33%
pcler8	131	100,00%	146	60,33%	96	39,67%			54	22,31%
cc	22	100,00%	7	100,00%					7	100,00%
count	104	100,00%	55	100,00%						
my_adder	17	100,00%	30	100,00%					30	100,00%
TOTALS	910	98,06%	892	73,42%	323	26,58%	18	**1,94%**	171	14,07%
			721	69,06%	323	**30,94%**				

Taula 5-8 *Resultats de l'heurística* List.

En la Taula 5-8 es poden veure els resultats de l'aplicació de l'heurística *List*. És evident que aquest tipus d'heurística és molt més conservadora que les tècniques basades en la predicció de la complexitat de l'agrupament resultant de la composició de funcions en l'equació (5-3). Poc més del 26% dels agrupaments no implementables són filtrats usant *List*, valor que correspon a poc menys del 31% dels agrupaments que podrien ser efectivament filtrats per aquesta estratègia. Per altre banda, cal destacar que menys del 2% dels agrupaments correctes són erròniament descartats. En realitat, aquest valor s'hauria de reduir pràcticament fins a zero. Efectivament, els agrupaments erròniament descartats corresponen a casos on la composició d'agrupaments bàsics excedeix els límits tecnològics, però composicions amb grups derivats d'aquests resulten implementables. Això només pot passar pel fet que la composició d'agrupaments derivats generi una funció més simple que la composició de grups bàsics, fet que implica l'existència de redundància en l'estructura de la Xarxa Booleana. Per tant l'aplicació de l'heurística després d'una

fase d'optimització de la Xarxa Booleana disminuirà el número d'agrupaments descartats erròniament.

Circuit	Acceptats		No filtrats		Filtrats		Perduts		No Pos.	
cm82a	2	100,00%	2	100,00%					2	100,00%
cm151a	34	94,44%	10	58,82%	7	41,18%	2	5,56%	7	41,18%
parity	14	100,00%	19	76,00%	6	24,00%			7	28,00%
cmb	4	100,00%	8	80,00%	2	20,00%			8	80,00%
cm163a	8	100,00%	14	73,68%	5	26,32%			13	68,42%
mux	1	100,00%	4	100,00%					4	100,00%
cm162a	31	100,00%	32	36,78%	55	63,22%			26	29,89%
cm150a	54	91,53%	48	48,48%	51	51,52%	5	8,47%	38	38,38%
cm85a	411	88,39%	283	46,62%	324	53,38%	54	11,61%	207	34,10%
cu	10	100,00%	7	87,50%	1	12,50%			6	75,00%
pm1	24	100,00%	3	100,00%					1	33,33%
pcler8	131	100,00%	146	60,33%	96	39,67%			54	22,31%
cc	22	100,00%	7	100,00%					7	100,00%
count	104	100,00%	55	100,00%						
my_adder	17	100,00%	30	100,00%					30	100,00%
TOTALS	867	93,43%	668	54,98%	547	45,02%	61	**6,57%**	410	33,74%
			258	32,05%	547	**67,95%**				

Taula 5-9 *Resultats de l'heurística* Hierarchy.

L'aplicació de l'heurística *Hierarchy* produeix els resultats que es poden veure en la Taula 5-9. El fet de classificar els agrupaments segons una certa jerarquia basada en els orígens de la seva creació, permet un filtrat més efectiu dels mateixos. D'aquesta manera, un 45% dels agrupaments no implementables són correctament filtrats. Aquest percentatge correspon a gairebé el 68% de tots els agrupaments susceptibles de ser filtrats per l'heurística. Finalment, els agrupaments erròniament descartats resten per sota del 7%. Com en el cas de *List*, és d'esperar que una optimització prèvia de la Xarxa Booleana que elimini les redundàncies farà baixar aquest percentatge. Cal destacar que aquesta tècnica permet un filtrat considerable d'agrupaments no implementables, mentre que el cost associat a pèrdua d'agrupaments correctes es manté en uns nivells relativament baixos.

En les taules que es troben a continuació es mostren els resultats obtinguts per l'aplicació de les heurístiques proposades en els circuits de prova. Aquest cop, el filtrat s'ha fet efectiu. D'aquesta manera es pot comprovar quin és l'impacte real de l'aplicació de l'heurística sobre el circuit, especialment pel que fa a la pèrdua d'agrupaments implementables. Efectivament, la pèrdua d'agrupaments implementables en un node pot provocar la pèrdua d'altres agrupaments implementables que potser derivarien d'aquest. En la columna *Total* es mostren el total d'agrupaments presents en el circuit quan no hi ha filtrat. La columna *Total Real* indica el número total d'agrupaments visibles en el circuit quan s'aplica l'heurística de filtrat. En la columna *Impl.* s'hi troben el número d'agrupaments implementables presents en el circuit. La columna *Acceptats* indica quants d'aquests són generats i acceptats usant l'heurística en estudi. La columna *No filtrats* conté el número d'agrupaments visibles per l'heurística que no són filtrats i resulten ser no implementables. Finalment, la

columna Filtrats conté el nombre d'agrupaments filtrats, correctament o no, per l'heurística proposada. La informació més interessant que es pot extreure d'aquestes taules és la reducció en el número total d'agrupaments, que apareix en percentatge sota la columna *Total Real*, el número total d'agrupaments implementables que passen el filtre i són acceptats i el número d'agrupaments que són filtrats per l'heurística.

Circuit	Total	Total Real		Implem.	Acceptats		No filtrats	Filtrats
cm82a	4	4	100,00%	2	2	100,00%	2	
cm151a	53	47	88,68%	36	25	69,44%	1	44,68%
parity	39	39	100,00%	14	14	100,00%	25	
cmb	14	14	100,00%	4	4	100,00%	7	21,43%
cm163a	27	27	100,00%	8	8	100,00%	16	11,11%
mux	5	5	100,00%	1	1	100,00%	4	
cm162a	118	102	86,44%	31	27	87,10%	60	14,71%
cm150a	158	144	91,14%	59	49	83,05%	3	63,89%
cm85a	1072	856	79,85%	465	242	52,04%	153	53,86%
cu	18	18	100,00%	10	10	100,00%	6	11,11%
pm1	27	27	100,00%	24	22	91,67%	1	14,81%
pcler8	373	218	58,45%	131	64	48,85%	26	58,72%
cc	29	29	100,00%	22	22	100,00%	7	
count	159	149	93,71%	104	99	95,19%	25	16,78%
my_adder	47	47	100,00%	17	17	100,00%	30	
TOTALS	2143	1726	**19,46%**	928	606	**65,30%**	366	**43,68%**

Taula 5-10 *Valors reals obtinguts usant l'heurística* MaxSerial.

Els resultats reals de l'aplicació de l'heurística *MaxSerial* es poden veure en la Taula 5-10. Més del 19% dels agrupaments ja no són generats. De la resta, prop del 44% són filtrats. Dels possibles agrupaments implementables presents en els circuits, el 34.70% són erròniament descartats o simplement no generats. De totes maneres, si comparem aquest valor amb el 34.70% d'agrupaments erròniament descartats per l'heurística en la Taula 5-5, podem veure els agrupaments descartats erròniament no intervenen en la generació de nous agrupaments implementables. Aquest fet és important, ja que si no fos així, els efectes de descartar un agrupament erroni es propagarien al llarg del circuit. L'explicació de la no existència d'aquest efecte de propagació es troba en el fet que els agrupaments que són erròniament descartats són, en general, prou complexes i difícilment es derivarà d'ells un nou agrupament implementable.

Circuit	Total	Tot.Real		Implem.	Acceptats		No filtrats	Filtrats
cm82a	4	4	100,00%	2	2	100,00%	2	0,00%
cm151a	53	49	92,45%	36	32	88,89%		34,69%
parity	39	39	100,00%	14	14	100,00%	19	15,38%
cmb	14	14	100,00%	4	4	100,00%	6	28,57%
cm163a	27	27	100,00%	8	8	100,00%	16	11,11%
mux	5	5	100,00%	1	1	100,00%		80,00%
cm162a	118	94	79,66%	31	25	80,65%	48	22,34%
cm150a	158	130	82,28%	59	47	79,66%		63,85%
cm85a	1072	782	72,95%	465	163	35,05%	78	69,18%
cu	18	18	100,00%	10	10	100,00%	3	27,78%
pm1	27	27	100,00%	24	22	91,67%	3	7,41%
pcler8	373	207	55,50%	131	72	54,96%	12	59,42%
cc	29	29	100,00%	22	15	68,18%	6	27,59%
count	159	159	100,00%	104	104	100,00%	25	18,87%
my_adder	47	47	100,00%	17	17	100,00%	15	31,91%
TOTALS	2143	1631	**23,89%**	928	536	**57,76%**	233	**52,85%**

Taula 5-11 *Valors reals obtinguts usant l'heurística* AddSerial.

En la Taula 5-11 es poden veure els resultats reals obtinguts en aplicar l'heurística *AddSerial* als circuits de prova. Prop del 24% dels agrupaments no són generats en el circuit degut a l'aplicació del filtre. De la resta, gairebé un 53% són filtrats. Aquests dos valors ens mesuren la capacitat de l'heurística d'accelerar el procés de generació d'agrupaments. Per altre banda, cal notar que un 42.24% dels agrupaments implementables en el circuit són filtrats o no generats. Si observem la Taula 5-6 veurem que els agrupaments erròniament descartats eren d'un 41.70%. Això indica que una petita part dels agrupaments descartats podien donar lloc a agrupaments implementables.

Circuit	Total	Tot.Real		Implem.	Acceptats		No filtrats	Filtrats
cm82a	4	4	100,00%	2	2	100,00%	2	0,00%
cm151a	53	26	49,06%	36	14	38,89%		46,15%
parity	39	39	100,00%	14	14	100,00%	25	0,00%
cmb	14	14	100,00%	4	4	100,00%	5	35,71%
cm163a	27	27	100,00%	8	8	100,00%	18	3,70%
mux	5	5	100,00%	1		0,00%		100,00%
cm162a	118	118	100,00%	31	31	100,00%	54	27,97%
cm150a	158	52	32,91%	59	24	40,68%		53,85%
cm85a	1072	1039	96,92%	465	215	46,24%	74	72,18%
cu	18	17	94,44%	10	9	90,00%	5	17,65%
pm1	27	27	100,00%	24	23	95,83%	1	11,11%
pcler8	373	278	74,53%	131	97	74,05%	25	56,12%
cc	29	29	100,00%	22	22	100,00%		24,14%
count	159	159	100,00%	104	104	100,00%	33	13,84%
my_adder	47	47	100,00%	17	17	100,00%	30	0,00%
TOTALS	2143	1881	**12,23%**	928	584	**62,93%**	272	**54,49%**

Taula 5-12 *Valors reals de l'heu´ristica* IncSupport.

En la Taula 5-12 es poden veure els resultats reals obtinguts en aplicar l'heurística *IncSupport* als circuits de prova. Poc més del 12% dels agrupaments no són generats en el circuit degut a l'aplicació del filtre. De la resta, un 54% són filtrats. Aquests dos valors ens mesuren la capacitat de l'heurística d'accelerar el procés de generació d'agrupaments. Per altre banda, cal notar que un 37.07% dels agrupaments implementables en el circuit són filtrats o no generats. Si observem la Taula 5-7 veurem que els agrupaments erròniament descartats eren d'un 34.70%. Això indica que una petita part dels agrupaments descartats podien donar lloc a agrupaments implementables.

Circuit	Total	Tot.Real		Implem.	Acceptats		No filtrats	No Pos.	Filtrats
cm82a	4	4	100,00%	2	2	100,00%		2	
cm151a	53	53	100,00%	36	36	100,00%	13	1	5,66%
parity	39	39	100,00%	14	14	100,00%	12	7	15,38%
cmb	14	14	100,00%	4	4	100,00%		8	14,29%
cm163a	27	27	100,00%	8	8	100,00%	1	13	18,52%
mux	5	5	100,00%	1	1	100,00%		4	
cm162a	118	118	100,00%	31	31	100,00%	38	10	33,05%
cm150a	158	158	100,00%	59	59	100,00%	77	3	12,03%
cm85a	1072	874	81,53%	465	397	85,38%	282	25	19,45%
cu	18	18	100,00%	10	10	100,00%	1	6	5,56%
pm1	27	27	100,00%	24	24	100,00%	2	1	
pcler8	373	373	100,00%	131	131	100,00%	92	54	25,74%
cc	29	29	100,00%	22	22	100,00%		7	
count	159	159	100,00%	104	104	100,00%	55		
my_adder	47	47	100,00%	17	17	100,00%		30	
TOTALS	2143	1945	**9,24%**	928	860	**92,67%**	573	171	**17,53%**

Taula 5-13 *Valors reals obtinguts usant l'heurística* List.

En la Taula 5-13 es mostren els resultats real de l'aplicació de l'heurística *List*. Poc més del 9% dels agrupament no són generats en el circuit a l'aplicar l'heurística. De la resta, poc més d'un 17% seran filtrats. Cal comentar que poc menys del 8% dels agrupaments implementables són no generats o descartats per l'heurística. Aquest número suposa un increment notable respecte el valor de menys del 2% de la Taula 5-8. Aquest increment es pot explicar pel fet que l'aplicació d'aquesta heurística elimina tots els possibles agrupaments d'un grup bàsic que resulta no implementable. Si alguns dels agrupaments descartats per l'heurística eren de complexitat baixa, un conjunt d'agrupaments derivats d'aquest no serà generat. Sortosament, però, aquesta situació no s'hauria de donar en circuits on la Xarxa Booleana ha estat optimitzada prèviament al mapatge tecnològic.

Circuit	Total	Tot.Real		Implem.	Acceptats		No filtrats	No Pos.	Filtrats
cm82a	4	4	100,00%	2	2	100,00%		2	
cm151a	53	51	96,23%	36	32	88,89%	6	7	25,49%
parity	39	39	100,00%	14	14	100,00%	12	7	33,33%
cmb	14	14	100,00%	4	4	100,00%		8	71,43%
cm163a	27	27	100,00%	8	8	100,00%	1	13	66,67%
mux	5	5	100,00%	1	1	100,00%		4	80,00%
cm162a	118	118	100,00%	31	31	100,00%	10	26	65,25%
cm150a	158	124	78,48%	59	51	86,44%	17	30	45,16%
cm85a	1072	841	78,45%	465	371	79,78%	66	155	48,04%
cu	18	18	100,00%	10	10	100,00%	1	6	38,89%
pm1	27	27	100,00%	24	24	100,00%	2	1	3,70%
pcler8	373	373	100,00%	131	131	100,00%	92	54	40,21%
cc	29	29	100,00%	22	22	100,00%		7	24,14%
count	159	159	100,00%	104	104	100,00%	55		
my_adder	47	47	100,00%	17	17	100,00%		30	63,83%
TOTALS	2143	1876	**12,46%**	928	822	**88,58%**	262	350	**42,22%**

Taula 5-14 *Resultats reals de l'heurística* Hierarchy. *Poc més del 12% dels agrupaments no són generats. De la resta, un xic més del 42% seran filtrats. Finalment, un xic menys del 12% dels agrupaments implementables són filtrats o no generats.*

En la Taula 5-14 es mostren els resultats real de l'aplicació de l'heurística *Hierarchy*. Poc més del 12% dels agrupament no són generats en el circuit a l'aplicar l'heurística. De la resta, poc més d'un 42% seran filtrats. Cal comentar que poc més del 11% dels agrupaments implementables són no generats o descartats per l'heurística. Aquest número suposa un increment notable respecte el valor de menys del 7% de la Taula 5-9. A l'igual que passava amb l'heurística *List*, aquest increment es pot explicar pel fet que l'aplicació d'aquesta heurística elimina tots els possibles agrupaments d'un grup bàsic que resulta no implementable.

Conclusions

En aquest capítol s'ha estudiat en detall el procés de generació de tots els agrupaments implementables pels nodes d'una Xarxa Booleana. S'ha descrit el problema i s'ha fet un estudi de la complexitat associada al mateix, que ha resultat ser molt elevada en el cas de considerar un mapatge tecnològic orientat a generadors de mòduls. En el cas del mapatge tecnològic orientat a biblioteques de cel·les éstàndard la complexitat del procés de generació d'agrupaments es veu reduïda pel fet de tenir una biblioteca limitada de cel·les que poden implementar les funcions associades als agrupaments. Això permet evitar la generació d'agrupaments complexes quan el suport de l'agrupament resultaria excedir el número màxim d'entrades de les cel·les de la biblioteca. Per la resta d'agrupaments, caldria comprovar si resulten implementables o no, mitjançant algoritmes d'aparellament.

La situació descrita anteriorment canvia radicalment quan el procés de generació d'agrupaments considera un mapatge tecnològic orientat a generadors de mòduls. En

aquest cas, el número de funcions que poden ser implementades és molt més elevat, com també ho és el número màxim d'entrades permeses en una porta. Així, el número d'agrupaments que pot presentar un circuit creix exponencialment. A més, el número d'agrupaments que cal generar i que després resulten ser no implementables, i per tant són descartats, supera el 50% del total d'agrupaments. Aquest fet fa pensar en l'aplicació d'heurístiques que permetin simplement no generar aquests agrupaments. En aquest sentit, s'han proposat dos grans tipus d'heurístiques. En primer lloc, aquelles que es basen en predir la complexitat dels agrupaments abans de generar-los. Aquesta tècnica és capaç d'eliminar un elevat percentatge dels agrupaments no implementables, però també elimina un conjunt considerable d'agrupaments que serien realment implementables. En segon lloc, s'han proposat heurístiques basades en la generació jeràrquica dels agrupaments. Aquestes són molt més conservadores i redueixen en gran mesura el percentatge d'agrupaments implementables eliminats erròniament, a costa d'aconseguir filtrar menys agrupaments no implementables. Cal dir que aquest tipus de tècnica resulta ser sensible a l'estructura de la Xarxa Booleana, pel que és molt més efectiva quan s'aplica sobre una Xarxa Booleana optimitzada.

L'existència de diferents heurístiques de filtrat d'agrupaments no implementables permet establir compromisos entre la velocitat desitjada en el procés de generació d'agrupaments i la qualitat del resultat final, que serà utilitzat a continuació pel procés de mapatge tecnològic. Al nostre entendre, l'heurística *Hierarchy* presenta un bon compromís entre efectivitat de filtrat i qualitat del resultat final, especialment quan la Xarxa Booleana ha estat prèviament optimitzada.

Com a conclusió, podríem destacar que, evidentment, les tècniques de filtrat d'agrupaments no implementables proposades en aquest capítol poden ser també aplicades en el cas de realitzar-se un mapatge tecnològic clàssic basat en biblioteques de cel·les. El fet que, segons el nostre coneixement, tècniques d'aquest estil no hagin estat usades anteriorment es veu explicat simplement per la menor complexitat del procés de generació d'agrupaments en el cas del mapatge tecnològic orientat a biblioteques de cel·les estàndard.