UNIVERSITATEA TEHNICÃ CLUJ-NAPOCA
FACULTATEA DE AUTOMATICÃ ŞI CALCULATOARE
CATEDRA DE CALCULATOARE

ing. Baruch Zoltan Francisc

CONTRIBUØII LA PROIECTAREA ASISTATÃ DE CALCULATOR A SISTEMELOR NUMERICE

Rezumatul tezei de doctorat

Conducãtor ştiinøific
Prof. dr. ing. PUSZTAI Kalman

Cluj-Napoca, 1998

CUPRINS

1. INTRODUCERE

1.1 Proiectarea sistemelor numerice

Proiectarea sistemelor numerice complexe nu este posibilã fãrã utilizarea sistemelor CAD în timpul tuturor fazelor procesului de proiectare. Majoritatea sistemelor numerice actuale sunt implementate sub forma circuitelor integrate LSI sau VLSI. Pe mãsura evoluøiei tehnologice de la integrarea pe scarã redusã (SSI sau MSI) la integrarea pe scarã largã, cerinøele impuse sistemelor CAD au crescut substanøial. Modificãrile tehnologice au schimbat de asemenea metodologia de proiectare. De exemplu, în cazul tehnologiei VLSI nu este foarte importantã reducerea numãrului de tranzistoare, deoarece reducerea costului prin minimizare logicã nu este semnificativã atunci când numãrul total de tranzistoare este de ordinul milioanelor. În schimb, este importantã reducerea costului interconexiunilor.

Existã diferite tipuri de circuite VLSI utilizate actualmente. Se pot distinge urmãtoarele categorii importante de circuite VLSI:

- Reøele de porøi
- Celule standard
- Macro-celule (blocuri constructive)
- Reøele logice programabile (PLA - *Programmable Logic Array*)
- Dispozitive logice programabile (PLD - *Programmable Logic Device*)
- Reøele de porøi programabile (FPGA - *Field-Programmable Gate Array*)

Dintre aceste tipuri de circuite utilizate pentru implementarea sistemelor numerice, în cadrul tezei se studiazã în primul rând circuitele FPGA. Motivul principal este cã aceste circuite sunt studiate într-o mãsurã mai redusã în literaturã, în cadrul tezei urmãrindu-se modul în care metodele generale de proiectare utilizate pentru circuitele VLSI în general pot fi utilizate şi pentru circuitele FPGA, şi care sunt metodele specifice a cãror utilizare se impune pentru circuitele FPGA.

1.2 Circuite FPGA

Circuitele FPGA (*Field-Programmable Gate Array*) sunt circuite integrate programabile de cãtre utilizator care permit un acces rapid la circuite VLSI configurabile. Un circuit FPGA constã dintr-o reøea de celule logice care pot fi interconectate prin comutatoare de rutare programabile. Circuitele FPGA combinã facilitãøile reøelelor de porøi programabile prin mãşti MPGA (*Mask Programmable Gate Array*) şi a dispozitivelor logice programabile PLD (*Programmable Logic Device*). De la circuitele MPGA s-a adoptat structura reøelei bidimensionale de celule logice, iar de la circuitele PLD s-a preluat programabilitatea de cãtre utilizator. Implementãrile din cadrul tezei de faøã au fost realizate pentru circuite FPGA.

Dupã introducerea lor de cãtre firma Xilinx [177], circuitele FPGA au evoluat în mod considerabil pe mãsurã ce au fost dezvoltate diferite noi tipuri de dispozitive [6], [7], [26], [27], [30], [157], [160], [178]. Utilizarea circuitelor FPGA s-a rãspândit pe scarã largã, ceea ce se datoreazã duratei reduse de producøie şi costului relativ redus al acestor dispozitive programabile. Faøã de tehnologiile alternative, circuitele FPGA oferã avantajul cã permit o reducere semnificativã a ciclului de proiectare şi asigurã o reducere a costului de producøie al circuitelor VLSI. Totuşi, programabilitatea de cãtre utilizator are şi dezavantaje: densitatea logicii şi performanøele de vitezã ale circuitelor FPGA sunt considerabil mai reduse decât ale celorlalte alternative. Deşi îmbunãtãøirile

din ultimii ani au permis creşterea performanøelor circuitelor, sunt necesare încã eforturi de cercetare pentru a dezvolta arhitecturi optime pentru circuitele FPGA.

1.3 Etapele de proiectare cu circuite FPGA

Proiectarea sistemelor numerice utilizând circuite FPGA este un proces complex, care necesitã resurse computaøionale importante. Pentru a se reduce complexitatea combinatorialã a acestei probleme, procesul de proiectare este împãrøit în mod obişnuit în urmãtoarele etape generale.

1) *Partiøionarea.* Sistemul proiectat, care de multe ori nu poate fi implementat într-un singur circuit FPGA, trebuie divizat în mai multe pãrøi, astfel încât fiecare parte sã poatã fi implementatã într-un singur circuit FPGA, şi sã poatã fi gestionatã independent de celelalte. Partiøionarea reprezintã în acelaşi timp o metodã algoritmicã pentru rezolvarea problemelor complexe de optimizare care apar în sinteza logicã sau în proiectarea fizicã a circuitelor VLSI.

2) *Maparea tehnologicã.* Pentru fiecare porøiune a sistemului care va fi implementatã într-un singur circuit FPGA, logica trebuie divizatã suplimentar în fragmente, astfel încât fiecare fragment sã aibã o dimensiune suficient de micã pentru a putea fi implementatã într-un singur bloc logic al circuitului. Aceastã divizare se realizeazã în cadrul etapei de mapare tehnologicã. Maparea tehnologicã este operaøia de transformare a unei reprezentãri logice cu nivele multiple într-o interconexiune de elemente logice dintr-o bibliotecã datã de elemente. Calitatea circuitelor sintetizate depinde în mare mãsurã de aceastã etapã.

3) *Plasarea.* În cadrul plasării, fiecãrui fragment care va fi implementat într-un bloc logic trebuie sã i se asigneze un bloc liber din cadrul circuitului. Pentru plasare trebuie minimizate anumite funcøii obiectiv, cu condiøia respectării unor restricøii impuse de proiectant, de procesul de implementare sau de stilul de proiectare. Cea mai importantã funcøie obiectiv este lungimea totalã a conexiunilor, care reprezintã o metricã utilizatã pe scarã largã pentru aprecierea calitãøii plasării. Exemple de restricøii sunt evitarea suprapunerii celulelor sau cerinøa ca celulele sã fie plasate într-o anumitã suprafaøã rectangularã. O plasare este acceptabilã dacã se poate obøine o rutare completã a circuitului în cadrul suprafeøei date. În cadrul tezei obiectivul principal al plasării este cel al asigurării rutabilitãøii circuitului.

4) *Rutarea.* Fiind dat un set de celule şi porturile acestora, un set de conexiuni şi locaøiile celulelor (obøinute în urma procesului de plasare), rutarea constã în determinarea cãilor adecvate pentru interconexiunile dintre seturile de pini. Aceste cãi adecvate minimizeazã funcøia obiectiv datã, supusã unor restricøii. Restricøiile pot fi impuse de proiectant, de procesul de implementare, de tipul circuitului sau de stilul de proiectare. Ca exemple de funcøii obiectiv se pot aminti reducerea lungimii totale a interconexiunilor, sau evitarea problemelor datorate întârzierilor semnalelor.

1.4 Motivaţia tezei

Dintre tipurile de circuite VLSI care se utilizează pentru implementarea sistemelor numerice, în cadrul tezei se au în vedere în primul rând circuitele FPGA, circuite care au câştigat o popularitate deosebită în ultimii ani, având ca principale avantaje reducerea costurilor de prototipizare şi reducerea semnificativă a duratei ciclului de proiectare. În cadrul tezei se studiază metodele de proiectare fizică pentru circuitele VLSI în general, urmărindu-se modul în care aceste metode se pot adapta pentru circuitele FPGA, şi care sunt metodele specifice de proiectare care trebuie utilizate pentru aceste circuite.

Principala motivaøie a tezei o constituie *elaborarea unei metodologii de proiectare* asistatã de calculator a sistemelor numerice *pentru circuite FPGA cu resurse limi-*

tate de rutare. Obiectivul principal este deci rutabilitatea circuitului, şi îndeplinirea acestui obiectiv este urmãritã în trei etape importante ale procesului de proiectare fizicã: *partiøionare*, *plasare* şi *rutare*. Asigurarea acestui obiectiv este cea mai dificilã pentru categoria amintitã de circuite.

Algoritmii de partiøionare pe baza tãieturii minime sunt utilizaøi deseori în cadrul etapelor de proiectare pentru circuitele VLSI şi FPGA. O altã motivaøie a tezei o constituie *studierea eficienøei aplicãrii algoritmilor de partiøionare pe baza tãieturii minime pentru circuitele FPGA cu resurse limitate de rutare*. Lungimea totalã a interconexiunilor este componenta principalã a funcøiei de cost utilizate de algoritmii tradiøionali de partiøionare pe baza tãieturii minime. S-a dorit sã se determine în ce mãsurã aceastã metricã este adecvatã pentru aceastã categorie de circuite FPGA.

Una din motivaøiile tezei o reprezintã *elaborarea unor algoritmi de plasare* pentru circuitele FPGA *care au ca obiectiv asigurarea rutabilitãøii* circuitelor. S-a urmãrit în primul rând rezolvarea problemei de plasare prin metoda partiøionãrii pe baza tãieturii minime. S-a urmãrit sã se studieze *efectul utilizãrii diferitelor secvenøe de aplicare a liniilor de tãieturã*, pentru a determina în ce mãsurã secvenøele tradiøionale permit obøinerea unor rezultate corespunzãtoare pentru circuitele FPGA cu resurse limitate de rutare.

O altã motivaøie a tezei este *investigarea posibilitãøii de utilizare a unor metode neconvenøionale pentru rezolvarea problemelor de proiectare fizicã* pentru circuitele VLSI şi FPGA. Dintre aceste metode neconvenøionale, s-a avut în vedere utilizarea algoritmilor genetici. Acešti algoritmi emuleazã procesul natural al evoluøiei ca o modalitate de evoluare cãtre o soluøie optimã. Aplicarea algoritmilor genetici este motivatã de faptul cã aceştia se caracterizeazã printr-un paralelism intrinsec, utilizând o populaøie de soluøii, care le conferã un avantaj faøã de alte metode, cum este metoda cãlirii simulate [102], care utilizeazã o singurã soluøie.

O altã motivaøie a tezei o constituie *elaborarea unui algoritm de rutare care sã rezolve conflictele la resursele de rutare*, prin considerarea efectului pe care îl are rutarea unei conexiuni asupra celorlalte conexiuni. S-a dorit de asemenea optimizarea atât din punct de vedere al numãrului de celule logice utilizate în cadrul circuitului, cât şi al întârzierilor de rutare.

De asemenea, una din motivaøiile tezei a constituit-o *conceperea şi implementarea unui sistem CAD* pentru proiectarea sistemelor numerice utilizând circuitele FPGA din seria *Atmel* 6000, care sã integreze rezultatele obøinute la studiul etapelor de proiectare fizicã. Motivul pentru care implementarea s-a realizat pentru circuitele *Atmel* din seria 6000 este cã aceste circuite nu dispuneau de un asemenea sistem CAD, sistemul de dezvoltare existent conøinând doar un editor grafic pentru configurarea circuitelor. S-a dorit de asemenea includerea în acest sistem a unui program de mapare tehnologicã pentru circuitele FPGA *Atmel*, ca şi posibilitatea specificãrii sistemului numeric cu ajutorul unui limbaj de descriere, în particular limbajul *ABEL*.

1.5 Organizarea tezei

Capitolul 2 prezintã situaøia actualã în domeniul circuitelor VLSI în general şi FPGA în particular, şi descrie în mod succint procesul de proiectare al acestor circuite. Sunt trecute în revistã principalele tipuri de circuite VLSI: reøele de porøi, celule standard, macro-celule, circuite FPGA. Sunt descrise exemple reprezentative de arhitecturi ale unor circuite FPGA comerciale, fiind prezentatã pe scurt şi arhitectura de rutare a acestor circuite.

Capitolul 3 prezintã problema de partiøionare a circuitelor în general, accentul fiind pus pe partiøionarea circuitelor cu resurse limitate de rutare. Este definitã problema de partiøionare, fiind prezentate sintetic diferite metode de partiøionare întâlnite în literaturã pentru circuitele VLSI şi FPGA. În cadrul capitolului se propun doi algoritmi de bipartiøionare pentru circuitele FPGA cu resurse limitate de rutare. Primul al-

goritm se bazeazã pe metoda tãieturii minime, şi urmãreşte echilibrarea numãrului de conexiuni din cadrul partiøiilor. Al doilea este un algoritm genetic, având un obiectiv similar cu primul algoritm.

În capitolul 4 se studiazã plasarea modulelor cu obiectivul asigurãrii rutabilitãøii circuitelor. Se descriu diferite metode de plasare, fiind prezentate şi douã metode neconvenøionale: plasarea prin algoritmi paraleli şi plasarea prin reøele neuronale artificiale. În acest capitol se propun doi algoritmi de plasare pentru circuitele FPGA cu resurse limitate de rutare, primul fiind bazat pe algoritmul de partiøionare pe baza tãieturii minime propus în capitolul 3. Se propune de asemenea o secvenøã de aplicare a liniilor de tãieturã care este mai eficientã decât secvenøele tradiøionale. Al doilea algoritm propus este un algoritm genetic.

Capitolul 5 tratează problema de rutare a circuitelor, accentul punându-se pe circuitele cu resurse limitate de rutare. Se descriu sintetic diferite metode de rutare globalã şi metode de rutare detaliatã. Acestea din urmã au fost împãrøite în metode de rutare detaliatã generalã şi metode de rutare prin canale. Se prezintã apoi problema de rutare a circuitelor FPGA şi se descriu metode specifice pentru rutarea acestor circuite. În cadrul capitolului se propune un algoritm de rutare pentru circuitele FPGA *Atmel* din seria 6000.

În capitolul 6 se prezintã un sistem CAD conceput şi implementat pentru proiectarea sistemelor numerice utilizând circuitele FPGA din seria *Atmel* 6000, care integrează algoritmii propuşi în capitolele precedente. Specificaøia de intrare este reprezentatã de o descriere în limbajul *ABEL*. Aceastã descriere este compilatã într-un set de ecuaøii cu ajutorul compilatorului sistemului de dezvoltare *Easy-ABEL*. Din setul de ecuaøii se generează o reprezentare internã a sistemului numeric. Apoi, sistemul CAD executã etapele de mapare tehnologicã, plasare şi rutare, şi generează un fişier pentru configurarea circuitului FPGA.

Capitolul 7 prezintã sumarul tezei, contribuøiile tezei, şi indicã unele posibilitãøi de cercetare şi dezvoltare viitoare. În final sunt listate referinøele bibliografice.

2. ANALIZA SITUAŢIEI ACTUALE ÎN DOMENIUL CIRCUITELOR VLSI ŞI FPGA

2.1 Introducere

În acest capitol se prezintã în mod sintetic situaøia actualã în domeniul circuitelor VLSI şi FPGA şi al proiectãrii acestor circuite. Se descriu etapele de proiectare ale circuitelor VLSI în general, pentru a scoate în evidenøã locul etapei de proiectare fizicã în cadrul procesului de proiectare, etapele proiectãrii fizice reprezentând subiectul principal al tezei.

2.2 Procesul de proiectare al circuitelor VLSI

Deoarece complexitatea circuitelor VLSI este de ordinul milioanelor de tranzistoare, proiectarea unui circuit VLSI este o sarcinã complexã. Pentru a reduce complexitatea procesului de proiectare, se introduc mai multe nivele de abstractizare. Pe mãsurã ce procesul avansează de la nivelele superioare la cele inferioare de abstractizare, se introduc din ce în ce mai multe detalii despre noul proiect. Nivelele tipice de abstractizare şi etapele de proiectare corespunzãtoare sunt ilustrate în Figura 2.1. Dupã cum se indicã în aceastã figurã, proiectul trece de la etapa de specificaøie la cea de fabricaøie cu ajutorul diferitelor utilitare CAD.

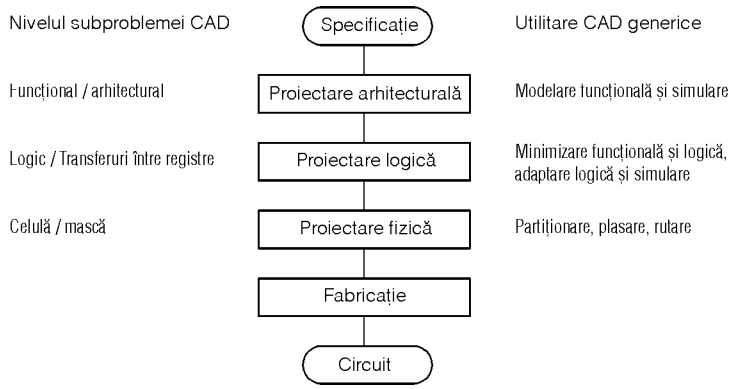

Figura 2.1. Nivele de abstractizare şi etapele corespunzãtoare de proiectare.

2.2.1 Proiectarea arhitecturală

Iniøial, proiectantul utilizează module de circuit ca unitãøi aritmetice, unitãøi de memorie, reøele de interconectare, controlere. Proiectarea unui circuit la acest nivel de abstractizare este numitã *proiectare arhitecturalã*. Deciziile luate în această etapã afectează în mod semnificativ costul şi performanøele proiectului.

Dupã definirea arhitecturii sistemului, este necesarã execuøia urmãtoarelor etape:

1. Proiectarea logicã detaliatã a modulelor de circuit;
2. Determinarea semnalelor de control necesare pentru activarea şi dezactivarea modulelor de circuit.

Prima etapã este numitã *proiectarea cãii de date*, iar etapa a doua *proiectarea cãii de control*. Fiind datã o specificaøie, obiectivul este de a se ajunge la un proiect care satisface toate constrângerile impuse de specificaøie, şi optimizează unul sau mai multe aspecte ale proiectului. Această problemã este numitã şi *sintezã hardware*. Pentru sinteza cãii de date şi a cãii de control au fost elaborate diferite programe de proiectare asistatã de calculator. Generarea automatã a cãii de date şi a cãii de control este numitã *sintezã de nivel înalt* [79] [121] [122].

2.2.2 Proiectarea logică

Dacã implementarea trebuie realizatã sub forma unui circuit VLSI utilizând componente aflate într-o *bibliotecã de module* (aceste module fiind numite şi *macrocelule*), următoarea etapã de proiectare este selectarea componentelor astfel încât sã se minimizeze costul total şi în acelaşi timp sã se maximizeze performanøele. Dupã procedura de selecøie, componentele (celulele) sunt amplasate pe suprafaøa de rutare şi sunt interconectate utilizând conexiuni de metal şi polisiliciu.

Dacã implementarea trebuie realizatã utilizând unul sau mai multe circuite FPGA, operaøiile efectuate în cadrul proiectării logice constau în principal din *partiøionare* şi *mapare tehnologicã*. În cadrul partiøionării, proiectul este divizat în mai multe pãrøi astfel încât sã fie posibilã implementarea fiecãreia într-un circuit FPGA. În cadrul mapării tehnologice, pentru fiecare parte care va fi implementatã într-un singur circuit FPGA, logica este divizatã în mai multe fragmente suficient de mici pentru a fi implementate într-un singur bloc logic al circuitului.

2.2.3 Proiectarea fizică

Proiectarea fizicã a unui circuit este etapa care precede fabricaøia acestuia. În modul cel mai general, proiectarea fizicã se referã la toate etapele de proiectare care urmeazã dupã proiectarea logicã şi care preced fabricaøia. Acestea cuprind toate sau o parte din urmãtoarele etape: partiøionare logicã, plasare şi rutare. Performanøele circuitului, din punct de vedere al spaøiului ocupat, al vitezei şi al fiabilitãøii, depind în mod critic de modul în care se realizează proiectarea fizicã.

Plasarea şi rutarea afectează în mod semnificativ suprafaøa ocupatã de circuit. Există două componente ale suprafeøei circuitului: suprafaøa funcøională şi cea de interconectare. Suprafaøa ocupatã de elementele active reprezintă suprafaøa funcøională. Interconexiunile utilizate pentru conectarea acestor elemente funcøionale contribuie la suprafaøa de interconectare. Interconexiunile lungi şi orificiile de trecere afectează nu numai performanøele, ci şi suprafaøa circuitului.

2.3 Tipuri de circuite VLSI

2.3.1 Rețele de porți

O reøea de porøi constă dintr-un număr mare de tranzistoare care sunt prefabricate sub forma unui tablou bidimensional. Iniøial tranzistoarele dintr-o reøea nu sunt conectate între ele. Pentru implementarea unui circuit printr-o reøea de porøi, trebuie plasate conexiuni metalice între tranzistoare utilizând procesul obişnuit de mascare. Acest proces de adãugare a conexiunilor metalice la o reøea de porøi este numit particularizare a reøelei. După acest proces, reøelele individuale de porøi pot fi separate, împachetate şi testate.

Deoarece toate etapele cu excepøia particularizării sunt identice pentru toate reøelele de porøi, indiferent de circuitul care va fi implementat, se poate stoca un număr mare de circuite care au fost prefabricate până la procesul de metalizare. Astfel, va fi necesar un timp foarte redus până la fabricarea circuitului final. Reøelele de porøi sunt numite şi reøele de porøi programabile prin măşti (*Mask Programmable Gate-Array - MPGA*). Costul producerii unui circuit cu o reøea de porøi este redus datorită randamentului ridicat. Aceasta deoarece există un număr redus de etape de prelucrare implicate într-o particularizare.

2.3.2 Celule standard

O celulă standard, numită şi policelulă, este un bloc logic care execută o funcøie standard. O bibliotecă de celule este o colecøie de informaøii legate de celulele standard. Informaøiile relevante despre o celulă constau din numele celulei, funcøionalitatea acestuia, aranjarea pinilor, şi amplasarea celulei pentru o anumită tehnologie. Celulele dintr-o anumită bibliotecă au aceeaşi înăløime.

Suprafaøa de amplasare este împărøită în mai multe rânduri. Un rând constă din celule plasate aproape unele de altele. Rândurile sunt separate prin canale de rutare orizontale. Celulele din acelaşi rând, sau cele din două rânduri alãturate pot fi interconectate prin intermediul canalului adiacent. Dacă trebuie conectate două celule aflate în rânduri non-adiacente, se utilizează celule de tip special, numite *celule de trecere*.

Faøă de reøelele de porøi, celulele standard oferă o flexibilitate mai mare. În cazul unui circuit cu celule standard, spaøiul de interconectare nu este fixat dinainte. Mai mult, celulele pot avea lãøimi diferite. Dezavantajul celulelor standard faøă de reøelele de porøi constă în faptul că pentru fabricaøia circuitului sunt necesare toate etapele de fabricaøie.

2.3.3 Macro-celule

Atât proiectarea cu reøele de porøi cât şi cea cu celule standard impun restricøii asupra celulelor care sunt utilizate. De exemplu, celulele standard trebuie să aibă aceeaşi înăløime. Dacă această restricøie este eliminată, celulele nu mai pot fi plasate pe rânduri. Circuitele la care pot varia ambele dimensiuni ale celulelor sunt numite circuite cu *macro-celule* sau *blocuri constructive*. Avantajul principal al acestora este că biblioteca poate conøine celule de o complexitate mult mai mare, de exemplu registre, unităøi aritmetice şi logice, memorii şi alte blocuri arhitecturale.

Există un avantaj important al memorării unor blocuri ca unităøi aritmetice şi logice într-o bibliotecă de celule. Asemenea blocuri pot fi proiectate astfel încât să aibă caracteristici de amplasare eficiente. Conceptul de memorare a celulelor într-o bibliotecă poate reduce în mod semnificativ efortul de proiectare. Există însă un dezavantaj al acestei metode. O bibliotecă de celule este dependentă de tehnologia de fabricaøie, deci sunt necesare biblioteci diferite pentru diferitele tehnologii. În cazul trecerii la o nouă tehnologie, este necesar un efort considerabil pentru reproiectarea celulelor.

2.3.4 Circuite FPGA

Similar unui circuit MPGA, un circuit FPGA (*Field Programmable Gate Array*) constă dintr-o reøea bidimensională de blocuri logice. De obicei, fiecare bloc logic poate fi programat pentru a implementa orice funcøie logică a intrărilor sale. Canalele şi blocurile de comutare dintre aceste blocuri conøin resurse de interconectare. Aceste resurse conøin de obicei segmente de interconectare de diferite lungimi. Interconexiunile conøin comutatoare programabile cu rolul de a conecta blocurile logice la segmentele de interconectare, sau un segment de interconectare la altul. În plus, există celule de I/E la periferia reøelei, care pot fi programate ca intrări sau ieşiri.

Avantajele circuitelor FPGA faøă de circuitele MPGA sunt costurile de prototipizare mai reduse şi durata mai scurtă de producøie. Principalele dezavantaje sunt viteza de operare mai redusă şi densitatea mai redusă a porøilor. Comutatoarele programabile şi circuitele de programare asociate necesită un spaøiu mai mare în cadrul circuitului comparativ cu conexiunile metalice din reøelele de porøi. Aceste comutatoare au de asemenea o rezistenøă şi capacitate semnificativă care determină o viteză redusă de operare.

2.4 Tipuri de circuite FPGA

Există două categorii principale de circuite FPGA: circuite cu memorii SRAM şi circuite cu antifuzibile.

Circuite cu memorii SRAM. Programarea acestor circuite se realizează prin celule de memorie statică. Logica este implementată cu ajutorul unor tabele (lookup table) realizate din celulele de memorie, intrările funcøiilor controlând liniile de adresă. Fiecare tabelă de 2^n celule de memorie implementează orice funcøie cu n intrări. Una sau mai multe tabele, combinate cu bistabile, formează un bloc logic configurabil. Aceste blocuri sunt aranjate într-un tablou bidimensional, segmentele de interconectare formând canale, similar cu reøelele de porøi. Segmentele se conectează la pinii blocurile logice din canale şi la alte segmente din blocurile de comutare prin intermediul tranzistoarelor de trecere controlate de celule ale memoriei de configurare.

Din această categorie de circuite FPGA fac parte cele ale firmelor Xilinx, Altera, AT&T.

Circuite cu antifuzibile. Un antifuzibil este un dispozitiv cu două terminale care în mod normal se află în starea de înaltă impedanøă, iar atunci când este expus la o tensiune ridicată, trece în starea cu rezistenøă redusă (300-500 Ω). Antifuzibilele au

dimensiuni reduse, astfel încât o arhitecturã bazatã pe antifuzibile poate conøine sute de mii sau milioane de antifuzibile. Pentru simplificarea arhitecturii øi a programãrii, circuitele FPGA bazate pe antifuzibile constau de obicei din rânduri de elemente logice configurabile cu canale de interconectare între ele, ca øi reøelele de porøi tradiøionale. Un bloc logic poate fi programat prin conectarea pinilor sãi de intrare la valori fixe sau la reøele de interconectare.

Din categoria circuitelor FPGA cu antifuzibile fac parte circuitele firmelor Actel, Quicklogic, Cypress.

2.5 Dificultățile proiectării fizice

Proiectarea fizicã este o problemã complexã de optimizare, care implicã mai multe funcøii obiectiv, de exemplu suprafaøa ocupatã de circuit, lungimea interconexiunilor, numãrul orificiilor de trecere. Circuitul trebuie sã satisfacã toate constrângerile impuse de specificaøie. De exemplu, dacã tehnologia utilizatã este cea a reøelelor de porøi, existã o constrângere asupra spaøiului disponibil pentru interconexiuni.

Existã dificultãøi practice atunci când se încearcã satisfacerea tuturor cerinøelor specificate. În primul rând, este dificilã modelarea problemei de proiectare fizicã dacã existã un numãr mare de constângeri øi se dorește optimizarea unui numãr mare de funcøii obiectiv. Problema este dificilã øi pentru cã unele din aceste funcøii obiectiv se aflã în conflict cu altele.

Problema de proiectare fizicã este divizatã de obicei în mai multe subprobleme mai simple. O subdiviziune posibilã este urmãtoarea:

1. Partiøionarea circuitului;
2. Planul de amplasare øi definirea canalelor;
3. Plasarea celulelor;
4. Rutarea globalã;
5. Ordonarea canalelor;
6. Rutarea detaliatã a liniilor de alimentare øi masã;
7. Rutarea detaliatã a celorlalte conexiuni.

Toate subproblemele menøionate sunt probleme de optimizare cu constrângeri. O asemenea problemã constã din gãsirea unei soluøii fezabile care satisface un set specificat de constrângeri de proiectare øi optimizează o funcøie obiectiv datã. Aceste subprobleme sunt de obicei NP-complete. De aceea, nu se cunosc algoritmi eficienøi care pot gãsi soluøii optime ale acestor probleme. Pentru soluøionarea problemelor complexe de optimizare de dimensiuni mari se utilizează *tehnici euristice*.

2.6 Concluzii

În acest capitol s-a prezentat în mod sintetic situaøia actualã în domeniul circuitelor VLSI øi FPGA, øi a fost descris procesul de proiectare al acestor circuite. Acest proces este complex, øi pentru reducerea complexitãøii se introduc mai multe nivele de abstractizare. Au fost prezentate nivelele tipice de abstractizare øi etapele de proiectare corespunzãtoare: proiectarea *arhitecturalã*, proiectarea *logicã* øi proiectarea *fizicã*.

În cazul circuitelor VLSI, metodele de proiectare sunt elaborate în general pentru un anumit tip de circuit. Au fost prezentate principalele tipuri de circuite VLSI utilizate: *reøele de porøi*, *celule standard*, *macro-celule*, *circuite FPGA*.

Existã numeroase dificultãøi legate de proiectarea fizicã. Aceasta este o problemã complexã de optimizare, care implicã mai multe funcøii obiectiv. Circuitul trebuie sã satisfacã toate constrângerile impuse de specificaøie. Problema de proiectare fizicã este divizatã în mai multe subprobleme mai simple. Aceste subprobleme sunt de obicei NP-complete. Pentru soluøionarea acestora se utilizează tehnici euristice.

3. PARTIȚIONAREA PENTRU CIRCUITELE CU RESURSE LIMITATE DE RUTARE

3.1 Introducere

Partiøionarea este tehnica de divizare a unui circuit sau sistem într-o colecøie de pãrøi de dimensiune mai micã (componente). Pe de o parte, partiøionarea este o etapã de proiectare pentru divizarea unui sistem în mai multe pãrøi care pot fi implementate prin componente separate, iar pe de altã parte partiøionarea reprezintã o metodã algoritmicã pentru rezolvarea problemelor complexe de optimizare care apar în sinteza logicã sau în proiectarea fizicã a circuitelor VLSI.

3.2 Definirea problemei de partiționare

Problema de partiøionare a circuitelor se poate formula ca o problemã de partiøionare a grafurilor. Un model matematic standard al circuitelor asociazã un graf $G = (V, E)$ cu lista de conexiuni a circuitului, unde vârfurile din V reprezintã elemente de circuit (module), iar muchiile din E reprezintã interconexiuni. Vârfurile øi muchiile grafului G pot fi ponderate pentru a reflecta suprafaøa ocupatã de modul sau importanøa unei conexiuni.

Existã douã formulãri de bazã ale problemei de bipartiøionare a circuitelor. Acestea sunt urmãtoarele [86]:

- *Tãietura minimã*: Fiind dat graful $G = (V, E)$, se partiøioneazã V în subseturile disjuncte U øi W astfel încât $e(U, W)$, adicã numãrul muchiilor în $\{(u, w) \in E \mid u \in U \text{ øi } w \in W\}$, este minimizat.
- *Bisecøia de lãøime minimã*: Fiind dat graful $G = (V, E)$, se partiøioneazã V în subseturile disjuncte U øi W, cu $|U| = |W|$, astfel încât $e(U, W)$ este minimizat.

Problema mai generalã de partiøionare este cea în care se formeazã k subseturi disjuncte. Aceasta se numeøte partiøionare cu k cãi øi este definitã astfel [146]:

- Fiind dat un graf $G = (V, E)$, unde fiecare vârf $v \in V$ are o dimensiune $s(v)$, iar fiecare muchie $e \in E$ are o pondere $w(e)$, se divide setul V în k subseturi V_1, V_2, …, V_k, astfel încât sã se optimizeze o funcøie obiectiv, øinând cont de anumite restricøii.

3.3 Restricții

Fiind dat un set de noduri V interconectate prin muchiile e_1, …, e_n, existã numeroase restricøii care pot fi impuse unei probleme de partiøionare. De exemplu, se poate specifica gãsirea unei partiøii (V_1, …, V_k) care satisface una sau mai multe din urmãtoarele restricøii:

1. Se specificã o dimensiune $s(v)$ pentru fiecare $v \in V$ øi o limitã superioarã de dimensiune S, cerându-se ca dimensiunea fiecãrui nod V_i sã fie cel mult S.
2. Sunt specificate valorile întregi pozitive n_1, …, n_k, øi se cere ca numãrul de elemente din V_i sã fie n_i, deci $|V_i| = n_i$, pentru fiecare i = 1, …, k.
3. Se specificã o limitã P a numãrului de pini, øi se cere ca numãrul de pini din fiecare V_i sã fie cel mult P, deci $P_i \le P$, pentru fiecare i = 1, …, k.
4. Trebuie minimizatã valoarea tãieturii totale a tuturor muchiilor, øi anume $\sum_{i=1}^{n} c(e_i)$. Cantitatea $\sum_{i=1}^{n} c(e_i)$ este numitã øi dimensiunea tãieturii partiøiei.

Pentru o bipartiøie, dimensiunea tãieturii este egalã cu numãrul de conexiuni tãiate de partiøie.

5. Pentru fiecare muchie e_i este specificatã o pondere $w(e_i)$, cerându-se ca valoarea tãieturii totale ponderate a tuturor muchiilor, deci $\sum_{i=1}^{n} w(e_i)c(e_i)$, sã fie minimizatã. Cantitatea $\sum_{i=1}^{n} w(e_i)c(e_i)$ este numitã şi dimensiunea ponderatã a tãieturii partiøiei.

3.4 Prezentarea sintetică a metodelor de partiționare

Existã diferite tehnici euristice pentru a se genera soluøii aproximative ale problemei de partiøionare [98], [146]. Acestea se pot clasifica în algoritmi *deterministici* şi *stohastici* (probabilistici). Metodele de partiøionare pot fi clasificate de asemenea ca fiind *constructive* şi *iterative*. O metodã constructivã porneşte de la o anumitã componentã nucleu, selectând apoi alte componente pentru a fi adãugate la soluøia parøialã, pânã la obøinerea unei soluøii complete. O metodã iterativã are ca scop îmbunãtãøirea calitãøii unei soluøii existente de partiøionare, de exemplu reducerea costului. De multe ori, o asemenea metodã se aplicã pentru îmbunãtãøirea soluøiei generate de o metodã constructivã.

Gruparea este o tehnicã pentru a determina componentele puternic conectate ale unui graf. Pentru partiøionarea unor circuite conøinând milioane de module gruparea de jos în sus este combinatã adesea cu partiøionarea de sus în jos. O formulare care unificã ambele strategii a fost publicatã în [94]. Gruparea a fost aplicatã şi pentru optimizarea performanøelor [179], [184]. Structuri formate din setul tuturor nodurilor unui bloc combinaøional între o singurã ieşire şi intrãrile care conduc la aceastã ieşire, au fost aplicate la partiøionarea de jos în sus a circuitelor FPGA pentru cãi critice [23]. Compromisul între timpul de execuøie şi performanøã este investigat în [185]. În [109] se descrie o metodã de grupare în care informaøiile globale de conectivitate ale grafului sunt obøinute din proprietatea de grupare a metodei vectorilor proprii.

Tehnicile de programare matematicã sunt utilizate pentru optimizarea unei funcøii obiectiv sub restricøiile unor inegalitãøi. Pentru rezolvarea problemelor de partiøionare au fost aplicate programarea cuadraticã [139], programarea booleanã cuadraticã [155], programarea liniarã [116].

Metodele spectrale au fost propuse în ultimii ani [39], [40], [86], [109]. Pe baza matricii de adiacenøã a grafului, obiectivul tãieturii minime poate fi rescris ca un sistem de ecuaøii. Vectorul propriu al valorii proprii minime diferite de zero a matricii poate fi interpretat ca o plasare liniarã sau ordonare a nodurilor grafului. Această ordonare poate fi divizatã pentru a obøine o partiøionare a nodurilor. În literaturã au fost publicate numeroase modificări ale acestei metode de bazã, inclusiv utilizarea mai multor vectori proprii. În cazul partiøionării cu cãi multiple s-a demonstrat faptul cã odatã cu creşterea numărului de vectori proprii, creşte calitatea partiøionării.

Partiøionarea bazatã pe plasare este strâns legatã de metodele spectrale. Aceste metode minimizează o funcøie obiectiv cuadraticã. Pentru plasare, s-a arãtat cã minimizarea unui obiectiv liniar conduce la rezultate îmbunãtãøite. În [139] această observaøie a fost utilizatã pentru obøinerea unor partiøionări de calitate mai bunã. Această metodã bazatã pe plasare a fost de asemenea extinsã la partiøionarea cu cãi multiple cu aplicaøii la circuitele FPGA [138].

În cazul metodelor bazate pe fluxul în reøele, fluxul direcøionat al semnalelor poate fi utilizat pentru îmbunãtãøirea performanøelor sistemului. Au fost propuse diferite tipuri de formulări ale fluxului în reøele [67], [94], [95], [116], [180], [181], [184]. Toate acestea au în comun faptul cã este generat un model al grafului din lista de conexiuni direcøionatã pentru a determina un flux maxim care este echivalent cu o tãieturã minimã.

Metodele constructive stohastice nu au fost utilizate frecvent pentru soluøionarea problemelor de partiøionare. Exemple mai recente sunt [90] şi [184]. În [90] este determinatã o odonare liniarã prin selectarea aleatoare a unor noduri de început. Prin utilizarea programãrii dinamice ordonarea este divizatã în grupe. În [184] este propusã o metodã probabilisticã pentru a reduce complexitatea computaøionalã a algoritmului bazat pe flux.

Au fost publicate numeroase metode deterministice iterative. Acestea interschimbã în mod iterativ noduri sau perechi de noduri pentru a minimiza numãrul de muchii tãiate. Din acest motiv, acestea sunt indicate în mod colectiv ca algoritmi de tãieturã minimã. Cele mai multe din acestea sunt de tip greedy [105], [146], [189]. Acešti algoritmi diferã în mod semnificativ în alegerea funcøiei obiectiv utilizate. Un obiectiv îmbunãtãøit bazat pe calculul probabilistic al câştigurilor este introdus în [69]. Cele mai multe implementãri utilizeazã configuraøii iniøiale aleatoare multiple [175] pentru cãutarea adecvatã în spaøiul soluøiilor şi a obøine performanøe predictibile. În [50] a fost propusã o metodã de bipartiøionare cu performanøe stabile, care nu necesitã generarea unui mare numãr de configuraøii iniøiale.

Îmbunãtãøirea iterativã poate fi combinatã cu gruparea pentru a reduce complexitatea calculelor. Deoarece metodele iterative deterministice sunt sensibile la modul de alegere a partiøiei iniøiale, în [117] a fost propusã o metodã bazatã pe gradient pentru a elimina acest dezavantaj. Au fost propuse şi metode de partiøionare de tãieturã minimã bazate pe programarea cuadraticã.

Metodele stohastice de îmbunãtãøire iterativã utilizate în mod obişnuit sunt cãlirea simulatã şi evoluøia simulatã [76], [189]. Cel mai important avantaj al acestora este cã pot evita minimele locale. O evaluare experimentalã a bipartiøionãrii în [186] aratã cã prin cãlirea simulatã se obøin anumite avantaje în privinøa calitãøii soluøiei, dar timpul de calcul consumat este foarte mare.

3.4.1 Algoritmul Kernighan-Lin

Algoritmul Kernighan-Lin (KL) a fost elaborat iniøial pentru bisecøionarea grafurilor, şi a fost extins apoi pentru rezolvarea problemei de bisecøionare a circuitelor [189]. Problema de bipartiøionare este caracterizatã printr-o matrice de conectivitate C, care este o matrice pãtratã cu un numãr de linii egal cu numãrul de noduri ale grafului circuitului. Elementul c_{ij} reprezintã suma ponderilor muchiilor care conectează elementele i şi j. În cazul în care muchiile au ponderi unitare, c_{ij} indicã numãrul de muchii care conectează i şi j. Rezultatul algoritmului de partiøionare este o pereche de seturi (blocuri) A şi B astfel încât $|A| = |B| = n$, $A \cap B = \varnothing$, iar setul de tãieturã are o dimensiune cât mai micã. Această dimensiune este mãsuratã prin T,

$$T = \sum_{a \in A, b \in B} c_{ab} \tag{3.6}$$

Algoritmul porneşte de la o partiøie iniøialã astfel încât $|A| = |B| = n$ şi $A \cap B = \varnothing$. Fie $P^* = \{A^*, B^*\}$ partiøia optimã, iar $P = \{A, B\}$ partiøia curentã. Pentru a se obøine P^* din P, trebuie sã se interschimbe un subset $X \subseteq A$ cu un subset $Y \subseteq B$, astfel încât:

(1) $|X| = |Y|$
(2) $X = A \cap B^*$
(3) $Y = A^* \cap B$

Presupunem cã existã o partiøie iniøialã $\{A, B\}$ de câte n elemente fiecare. Kernighan şi Lin au elaborat o procedurã de tip greedy pentru a identifica douã subseturi $X \subseteq A$ şi $Y \subseteq B$, de cardinalitãøi egale, astfel încât în urma interschimbãrii acestora costul partiøiei este îmbunãtãøit. În cursul procedurii, se calculează câştigurile interschimbãrii oricãror douã module $a \in A$ şi $b \in B$. Este selectatã perechea (a_1, b_1) care conduce la câştigul maxim g_1, iar elementele a_1 şi b_1 sunt blocate pentru a nu fi luate

în considerare la urmãtoarele interschimbãri. Câştigurile sunt recalculate, iar apoi este selectatã şi blocatã a o doua pereche (a_2, b_2) cu câştigul maxim g_2. De notat cã g_2 este câştigul interschimbãrii a_2 cu b_2 dupã ce a_1 a fost deja interschimbar cu b_1. Astfel, câştigul interschimbãrii perechii (a_1, b_1) urmat de interschimbarea (a_2, b_2) este $G_2 = g_1 + g_2$. Procesul continuã selectând (a_1, b_1), (a_2, b_2),…,(a_i, b_i) … (a_n, b_n), câştigurile corespunzãtoare fiind g_1, g_2, …, g_i,…, g_n. În general, câştigul interschimbãrii primelor k perechi (a_1, b_1), (a_2, b_2),…,(a_k, b_k), $1 \le k \le n$ este $G_k = \sum_{i=1}^{k} g_i$. Dacã nu există un k astfel ca $G_k > 0$ atunci partiøia curentã nu poate fi îmbunãtãøitã; în caz contrar se alege valoarea k care maximizează G_k şi se efectuează interschimbarea {a_1, a_2,…, a_k} cu {b_1, b_2,…, b_k) în mod permanent.

Procedura de îmbunãtãøire a unei partiøii, indicatã mai sus, constituie un singur pas al algoritmului Kernighan-Lin. Partiøia obøinutã dupã pasul i constituie partiøia iniøialã pentru pasul $i + 1$. Iteraøiile se terminã atunci când $G_k \le 0$, deci nu se mai pot obøine îmbunãtãøiri suplimentare prin interschimbarea perechilor.

3.4.2 Variante ale algoritmului Kernighan-Lin

Algoritmul Kernighan-Lin poate fi extins pentru a rezolva şi alte cazuri ale problemei de partiøionare. O parte din acestea sunt urmãtoarele:

Blocuri cu dimensiuni inegale. În acest caz se partiøionează un graf $G = (V, E)$ cu $2n$ vârfuri în douã subgrafuri cu dimensiuni inegale n_1 şi n_2, $n_1 + n_2 = 2n$.

Elemente cu dimensiuni inegale. Se generează o bipartiøie a unui graf ale cãrui vârfuri au dimensiuni inegale.

Partiøionare cu k cãi. Se presupune cã graful are $k \cdot n$ vârfuri, $k > 2$, şi este necesarã generarea unei partiøii cu k cãi, fiecare cu n elemente.

Optimalitatea perechilor de partiøii este doar o condiøie necesarã a optimalitãøii în cazul problemei de partiøionare cu k cãi. Uneori va fi necesarã o interschimbare complexã de trei sau mai multe elemente din trei sau mai multe subseturi pentru a se obøine soluøia optimã globalã.

3.4.3 Euristica Fiduccia-Mattheyses

Fiduccia şi Mattheyses au elaborat o euristicã iterativã care ia în considerare atât conexiunile multipin, cât şi dimensiunile elementelor de circuit. Contribuøia acestora constã în permiterea mutãrii unui singur nod la un moment dat între cele douã subseturi ale partiøiei, o analizã şi o implementare atentã a efectului mutãrii unui singur nod asupra valorilor D ale altor noduri, şi utilizarea unei structuri de date eficiente pentru a se evita cãutarea inutilã a nodului care va fi mutat în etapa urmãtoare [189].

Euristica Fiduccia-Mattheyses este o tehnicã utilizatã pentru a gãsi soluøia la urmãtoarea problemã de bipartiøionare: Fiind dat un circuit constând din C celule conectate printr-un set de N conexiuni, problema este de a partiøiona circuitul C în douã blocuri A şi B astfel încât numãrul conexiunilor care au celule în ambele blocuri este minimizat şi este satisfãcut factorul de echilibru r [76]. Principalele diferenøe între euristicile Kernighan-Lin şi Fiduccia-Mattheyses sunt urmãtoarele [146]:

1. În cazul euristicii Fiduccia-Mattheyses este selectatã la un moment dat o singurã celulã, din oricare bloc, pentru a fi mutatã în blocul complementar.
2. Euristica Kernighan-Lin partiøionează un graf în douã blocuri astfel încât costul muchiilor tãiate este minim, în timp ce euristica Fiduccia-Mattheyses are ca scop reducerea costului conexiunilor tãiate de partiøie.

3. În locul câștigului datorat interschimbării a două celule este calculat câștigul datorat mutării unei singure celule dintr-un bloc în altul. Numărul total de celule care sunt mutate este dat de secvența cea mai bună de mutări c_1, c_2,…, c_k.

3.4.4 Partiționarea prin metoda călirii simulate

Călirea simulatã este o tehnicã iterativã utilizatã pe scarã largã pentru rezolvarea diferitelor probleme combinatoriale de optimizare. A fost utilizatã pentru majoritatea problemelor CAD, inclusiv pentru partiționare. Este o euristicã adaptivã care aparține clasei algoritmilor stohastici. Această euristicã a fost introdusã pentru prima datã de Kirkpatrick, Gelatt și Vecchi [102].

Euristica de călire simulatã se inspirã din procesul de rãcire controlatã a metalelor topite pentru a se obține o structurã cristalinã corespunzãtoare. Dacã se comparã optimizarea cu procesul de călire, se poate realiza o analogie între obținerea optimului global și obținerea structurii cristaline dorite.

Partea principalã a algoritmului de călire simulatã este procedura *Metropolis*, care simuleazã procesul de călire la o temperaturã datã *T*. Această procedurã constã în generarea unui lanț Markov de stări, ale cãror energii ar avea o distribuție Boltzmann dacã lanțul ar avea o lungime infinitã. Pe lângã temperaturã, procedura Metropolis mai are ca intrări soluția curentã *S* care va fi îmbunãtãțitã, și valoarea *M*, care este durata de timp în care este aplicatã călirea la temperatura *T*. Temperatura este redusã lent de la valoarea T_0 în progresie geometricã, în funcție de parametrul α. Durata de timp a procesului de călire la o anumitã temperaturã este mãritã gradat pe mãsurã ce temperatura este micșoratã. Aceasta se realizează utilizând parametrul $\beta > 0$.

3.4.5 Partiționarea prin tăietura proporțională

Fiind dat un circuit $N = (V, E)$, unde V este setul de noduri, iar E este setul de muchii, fie c_{ij} capacitatea unei muchii care conectează nodul i cu nodul j. (A, A') indicã o tãieturã care separã un set de noduri A de $A' = V - A$. Capacitatea acestei tãieturi este egalã cu $C_{AA'} = \sum_{i \in A} \sum_{j \in A'} c_{ij}$. Proporția acestei tãieturi este definitã prin: raportul

$$R_{AA'} = \frac{C_{AA'}}{|A| \times |A'|} \tag{3.17}$$

unde $|A|$ și $|A'|$ reprezintã cardinalitatea subseturilor A și A'. Această metricã a fost propusã de Wei și Cheng [175] și s-a dovedit o funcție obiectiv de succes pentru numeroase aplicații. Numărătorul reprezintã criteriul de tãieturã minimã, în timp ce numitorul favorizează o partiție echilibratã, deoarece $|A| \times |A'|$ este maxim atunci când $|A| = |A'|$. Tãietura proporționalã este tãietura care generează proporția minimã $R_{AA'}$ dintre toate tãieturile circuitului, deci $\min_A (C_{AA'} / |A| \times |A'|)$ ($A \subset V$ și $A \neq \emptyset$, $A' \neq \emptyset$).

Wei și Cheng [175] au elaborat o euristicã rapidã, care se bazează pe algoritmul Fiduccia-Mattheyses [76], datoritã eficienței acestuia. Algoritmul pentru tãietura proporționalã constã din trei faze principale: 1) inițializare, 2) deplasare iterativã, și 3) interschimbarea grupurilor.

3.4.6 Partiționarea cu performanțe stabile

Performanțele algoritmului Kernighan-Lin s-au dovedit a fi foarte dependente de alegerea partiției inițiale, iar rezultatele finale variazã în mod semnificativ ca o consecințã a diferitelor partiții de început [50]. Pentru a se evita blocarea în minime locale, este necesar un numãr mare de rulãri ale algoritmului asupra unor partiții inițiale generate aleator, proces care necesitã un timp ridicat. În plus, probabilitatea de a gãsi

soluøia optimã într-o singurã încercare scade exponenøial pe mãsurã ce dimensiunea circuitului creşte.

Cheng şi Wei [50] ajung la concluzia cã o tehnicã de grupare de sus în jos este esenøialã pentru o partiøionare eficientã. Pentru a reduce dimensiunea circuitelor, se utilizeazã o tehnicã de partiøionare recursivã de sus în jos, şi se divide întregul circuit în grupuri mici, puternic conectate. Astfel, grupurile generate la fiecare partiøionare se reduc ca dimensiune. Ultimul pas constã în rearanjarea grupurilor în douã subseturi care respectã restricøia de dimensiune. Deoarece numãrul grupurilor este relativ mic comparativ cu numãrul modulelor circuitului, se pot efectua numeroase încercãri ale operaøiei de rearanjare.

Circuitul este divizat mai întâi în grupuri mici, prin execuøia recursivã a algoritmului de tãieturã proporøionalã. Fiecare grup va conøine un numãr diferit de module de dimensiuni diferite. Aceste grupuri sunt rearanjate apoi în douã subseturi, respectându-se restricøiile de dimensiune. Se aplicã apoi algoritmul Fiduccia-Mattheyses circuitului contractat, în mod repetat, pentru a se obøine rezultate bune. În final, se execută din nou algoritmul Fiduccia-Mattheyses asupra circuitului original expandat.

3.4.7 Partiționarea prin metode spectrale

O clasã importantã de tehnici de partiøionare este reprezentatã de metodele "spectrale", care utilizeazã valori proprii şi vectori proprii ale matricilor obøinute din graful circuitului. Un circuit poate fi reprezentat ca un graf nedirecøionat $G = (V, E)$ cu $|V| = n$ vârfuri $v_1, \ldots, v_n$. G se poate reprezenta printr-o *matrice de adiacenøã* de $n \times n$ elemente $A = A(G)$, unde $A_{ij} = 1$ dacã $(v_i, v_j) \in E$ şi $A_{ij} = 0$ în caz contrar. Dacã G are muchii ponderate, A_{ij} este egal cu ponderea muchiei $(v_i, v_j) \in E$, şi prin convenøie $A_{ii} = 0$ pentru orice $i = 1, \ldots, n$. Dacã se notează cu $d(v_i)$ gradul vârfului v_i (suma ponderilor tuturor muchiilor incidente lui v_i), se obøine *matricea de grad*, notată cu D, care este o matrice diagonalã definitã prin $D_{ii} = d(v_i)$. Valorile proprii şi vectorii proprii ale unor asemenea matrici sunt studiate de un subdomeniu al teoriei grafurilor care se ocupã de spectrele grafurilor.

Hagen şi Kahng au arãtat în [86] legãtura teoretricã existentã între spectrele grafurilor şi tãieturile proporøionale optime. În teoria dezvoltatã se utilizează vectori proprii ale matricii $Q = D - A$, numitã Laplacian al lui G, unde D şi A sunt definite mai sus, matrice care a fost utilizatã şi de cãtre Hall [87]. Boppana, Donath şi Hoffman, ca şi aløi autori, utilizează matrici diferite derivate din graful circuitului, dar se bazează pe proprietãøi matematice similare pentru a elabora formulări bazate pe valori proprii şi a defini relaøia existentã cu partiøionarea.

Un numãr de autori au arãtat faptul cã gãsirea grupurilor naturale din cadrul circuitelor este utilã pentru mai multe aplicaøii, de exemplu: a) partiøionarea cu cãi multiple, în special atunci când dimensiunile partiøiilor nu sunt cunoscute dinainte; b) amplasarea constructivã a modulelor; c) situaøiile în care dimensiunea circuitului este foarte mare şi trebuie utilizatã gruparea pentru a reduce dimensiunea intrării asupra cãreia se aplicã partiøionarea. Aceste grupuri pot fi gãsite în mod eficient prin utilizarea metodei spectrale, deoarece al doilea vector propriu x conøine atât informaøii de partiøionare, cât şi de grupare. Hagen şi Kahng au arãtat cã o interpretare directã a celui de-al doilea vector propriu sortat poate identifica imediat grupurile naturale din cadrul grafului. Rezultatele obøinute sunt foarte bune mai ales pentru circuite de dimensiuni mari.

3.4.8 Partiționarea pe baza rețelelor de flux

Teorema fluxului maxim şi tãieturii minime (Ford şi Fulkerson) pentru reøele este o importantã tehnicã combinatoricã de optimizare. Aceasta are numeroase aplicaøii în proiectarea circuitelor VLSI, ca de exemplu plasarea liniarã sau maparea

tehnologicã pentru circuitele FPGA. Tehnica fluxului maxim şi tãieturii minime este o metodã naturalã de aflare a tãieturii minime într-un graf.

Yang şi Wong [180] au propus o metodã pentru modelarea exactã a unei liste de conexiuni (sau, în mod echivalent, a unui hipergraf) printr-o reøea de flux, şi o euristicã de bipartiøionare echilibratã bazatã pe utilizarea repetatã a tehnicii fluxului maxim şi tãieturii minime. Aceştia utilizează o noøiune de *bipartiøie r-echilibratã*, care este o bipartiøie astfel încât o componentã are o pondere care este o fracøiune r a ponderii totale W. În cazul special când $r = ½$, o bipartiøie r-echilibratã este o bipartiøie echilibratã. Deoarece în practicã nu este necesarã impunerea strictã a criteriului de r-echilibrare, se poate introduce un factor de deviere ε pentru a permite devierea ponderii unei componente de la $(1 - \varepsilon)\ rW$ la $(1 + \varepsilon)\ rW$.

3.4.9 Multipartiționarea

Multipartiøionarea sau partiøionarea cu cãi multiple este o extensie importantã a bipartiøionãrii deoarece asigurã un model natural şi de acurateøe mai mare pentru numeroase aplicaøii de partiøionare. Acest tip de partiøionare a fost abordat prin diferite metode: prin cãlire simulatã [105], prin utilizarea vectorilor proprii ale matricilor provenite din lista de conexiuni a circuitului [39], [86], [87], sau prin migrarea grupurilor [105], [155]. Acestea din urmã se concentrează asupra evaluãrii mutãrilor care determinã îmbunãtãøirea maximã a partiøiei curente.

Yeh *et al*. [183] au propus un algoritm de partiøionare cu îmbunãtãøire iterativã care utilizează un nou model pentru procesul de mutare. În plus, algoritmul poate utiliza diferite funcøii obiectiv cu aplicaøii în proiectarea circuitelor VLSI.

Un sistem este reprezentat printr-un hipergraf $H(V, E)$, unde $V = \{v_i \mid i = 1, 2, \ldots, n\}$ este setul nodurilor şi $E = \{e_u \mid u = 1, 2, \ldots, m\}$ este setul conexiunilor. Fiecare conexiune e_u este un subset al lui V cu cardinalitatea $|e_u| \geq 2$. Fiecare nod v_i are dimensiunea $s(v_i)$. $S(V) = \sum_{v \in V} s(V_i)$ reprezintã dimensiunea hipergrafului H. O partiøionare cu k cãi asignează nodurile lui V în k partiøii, cu fiecare partiøie b conøinând un subset nevid V_b al lui V. Se defineşte *acoperire* a unei conexiuni e ca fiind 0 dacã e aparøine unei singure partiøii, şi f dacã e conectează f partiøii, cu $f \geq 2$.

Euristica de partiøionare cu cãi multiple propusã de Yeh *et al*. se numeşte algoritmul *Primal-Dual* (PD). Algoritmul constã din trei faze: 1) partiøionare recursivã prin tãietura proporøionalã pentru formarea grupurilor; 2) iteraøie *Primal-Dual* asupra sistemului grupat; 3) iteraøie *Primal-Dual* asupra sistemului original.

3.4.10 Partiționarea prin metode probabilistice

Cele mai multe metode de partiøionare prin îmbunãtãøire iterativã calculează cãştigurile datorate mutãrii nodurilor pe baza informaøiilor locale din lista de conexiuni care urmãresc îmbunãtãøirea imediatã a tãieturii. Krishnamurthy [105] a sugerat o metodã de calcul a cãştigurilor pe baza unor informaøii mai globale, obøinându-se partiøii de calitate mai bunã. Necesarul de memorie al acestui algoritm este foarte ridicat. Niciuna din aceste metode nu poate însã prevedea cu acurateøe starea viitoare a unei conexiuni.

Dutt şi Deng [69], [70] au propus o metodã pe baze probabilistice pentru calculul cãştigurilor, care poate determina implicaøiile globale şi viitoare ale mutãrii unui nod în orice etapã a procesului de partiøionare. Fiecãrui nod u i se asociază o probabilitate $p(M(u))$ (prescurtatã ca $p(u)$) a evenimentului $M(u)$ ca u sã fie mutat efectiv în celãlalt subset în pasul curent al procesului de partiøionare. Din aceastã probabilitate se calculează cãştigurile probabilistice (sau potenøiale) $g(u)$ ale nodurilor, ceea ce

furnizează o indicaøie corectã a beneficiului care se obøine în urma mutării acestora în celãlalt subset.

Dupã ce s-au determinat probabilitãøile iniøiale, se calculează câştigurile probabilistice ale nodurilor. Din aceste câştiguri, se recalculează probabilitãøile nodurilor, şi din acestea se obøin câştiguri mai exacte ale nodurilor. Acest proces se repetă pentru un număr de iteraøii. Dupã terminarea acestui proces iniøial, nodurile cu câştigurile cele mai mari sunt mutate între cele două subseturi, ca şi în cazul altor metode iterative. Dupã fiecare mutare, se actualizează câştigurile şi probabilitãøile nodurilor. De asemenea, la fiecare mutare se calculează câştigul imediat obøinut. La sfârşitul etapei curente, se efectuează în mod efectiv mutările pânã în punctul în care se obøine un câştig imediat maxim.

3.5 Partiționarea pentru circuitele FPGA

Atunci când dimensiunea circuitului proiectat este prea mare pentru a fi configurat într-un singur circuit FPGA, circuitul trebuie partiøionat în subcircuite. Partiøionarea circuitelor FPGA multiple trebuie sã satisfacã restricøii suplimentare asupra dimensiunii subcircuitelor şi a numărului terminalelor de I/E. Pentru a øine cont de restricøiile suplimentare, au fost publicaøi un număr de algoritmi de partiøionare pentru circuitele FPGA [40], [89], [101], [176].

3.5.1 Partiționarea ierarhică pentru circuite FPGA multiple

Fie un circuit cu n noduri (sau blocuri CLB) şi fie V setul nodurilor, iar E setul conexiunilor, unde o conexiune $e \in E$ cuprinde două sau mai multe elemente din V. Problema de partiøionare pentru circuite FPGA multiple poate fi definitã formal dupã cum urmează. Fiind dat un hipergraf $H(V, E)$, o funcøie de cost f şi un set de I circuite FPGA cu restricøiile de dimensiune şi de pini $S_1, \ldots, S_I$, respectiv $P_1, \ldots, P_I$, trebuie sã se gãsească o mapare o: $V \rightarrow \{1, 2, \ldots, I\}$ astfel încât

i. $a_i \leq S_i$ pentru fiecare i = 1, 2, …, I, unde $a_i = \{|v \mid o(v) = i, v \in V\}|$. De notat cã dimensiunea fiecãrui nod (CLB) este 1.

ii. $b_i \leq P_i$ pentru fiecare i = 1, 2, …, I, unde b_i este numărul de conexiuni dintre un nod din subcircuitul i şi un alt nod din subcircuitul $j \neq i$.

iii. Funcøia de cost f este minimizatã.

Kim şi Shin [101] au elaborat un algoritm de partiøionare în care funcøia de cost include atât dimensiunea tãieturii, cât şi întârzierea introdusã de conexiune. Sunt estimate întârzierile pe cãile critice, şi se øine cont de acestea în timpul grupării şi partiøionării. Cea mai bunã cale de a minimiza întârzierile pe o cale criticã este de a se plasa toate nodurile cãii respective în acelaşi circuit. De aceea, se pot grupa mai multe elemente apropiate conectate, grupul respectiv fiind considerat ca un obiect în timpul primei faze a partiøionării. Mãsura de apropiere dintre două elemente (noduri) conectate este reprezentatã de o pondere a unei muchii.

Metoda de partiøionare încearcã sã minimizeze costul, satisfãcând în acelaşi timp restricøiile asupra numărului de terminale de I/E şi asupra dimensiunii circuitelor. Algoritmul de partiøionare constã din două faze. În prima fazã, cea de partiøionare iniøialã, se executã o partiøionare ierarhicã bazatã pe grupare, care este modificatã din [156]. Funcøia obiectiv a primei faze este suma ponderatã a dimensiunii tãieturii şi a întârzierii maxime. Pentru a elimina dependenøa de partiøia iniøialã, se partiøionează grupurile de *N1* ori pornind de la partiøia iniøialã generatã aleator, şi apoi se alege rezultatul cel mai bun. Acest rezultat este expandat şi optimizat din nou pentru a finaliza faza de partiøionare iniøialã. În faza a doua, cea de îmbunãtãøire a partiøiei, se îmbunã-

tãøeşte partiøia obøinutã în prima fazã pentru a satisface toate restricøiile datorate circuitelor FPGA date.

3.5.2 Partiționarea pentru circuite FPGA cu structuri de tip PLA

Maparea unor ecuaøii booleene într-un numãr minim de circuite este o sarcinã dificilã. O parte a acestei probleme o reprezintã partiøionarea logicii în numãrul minim de structuri PLA (*Programmable Logic Array*) de dimensiune fixã. Existã un numãr mic de lucrãri legate de partiøionarea logicii pentru arhitecturi de tip PLA. Hasan *et al.* [89] au elaborat o metodã pentru multipartiøionarea unei funcøii logice cu ieşiri multiple într-un numãr minim de subfuncøii pentru maparea în structuri PLA de dimensiune fixã sau blocuri funcøionale ale unui circuit FPGA.

Fiind dat un set de ecuaøii booleene, problema este de a le partiøiona într-un numãr minim de grupuri, astfel încât fiecare din acestea sã poatã fi implementat într-un bloc funcøional al circuitului. Obiectivul este de a se minimiza numãrul de blocuri funcøionale utilizate de mapare. Fiecãrei funcøii de ieşire i se asociazã un cost, reprezentând restricøiile pe care le impune pentru algoritmul de partiøionare. Restricøiile pentru blocurile funcøionale sunt numãrul limitat de intrãri şi termeni produs partajaøi de o ieşire. Costul se exprimã ca o combinaøie ponderatã a acestor restricøii.

Se începe prin efectuarea unei partiøionãri de tip greedy cu restricøii relaxate, pentru a se obøine blocurile partiøiei iniøiale. Restricøiile fiind relaxate, este posibil ca $O_i > O_m$ şi $P_i > P_m$, unde O_i şi P_i sunt parametrii care definesc dimensiunea blocului funcøional. Iniøial se alege un numãr mai mare de ieşiri decât cel maxim implementabil, şi se minimizeazã acel grup de ieşiri. Se aleg apoi O_m ieşiri din grupul minimizat. Blocurile partiøiei iniøiale conøin un numãr mai mare de termeni produs partajaøi. Se aşteaptã ca numãrul termenilor produs partajaøi sã scadã în timpul procedurii de minimizare.

Dupã procedura de partiøionare de tip greedy se executã o procedurã de rafinare. Aceasta are urmãtorii parametri de intrare: *O*, *I*, *P*, şi un bloc valid de ieşiri. Un bloc valid de ieşiri este cel care poate fi implementat într-un bloc funcøional al circuitului. Procedura se terminã dacã existã numãrul maxim de ieşiri permise în bloc sau dacã s-a încercat eliminarea tuturor ieşirilor cu excepøia celei mai costisitoare.

3.6 Metode neconvenționale de partiționare

3.6.1 Partiționarea prin evoluție stohastică

Metodologia de evoluøie stohasticã (*Stochastic Evolution* - *SE*) utilizeazã analogia dintre tehnicile de îmbunãtãøire iterativã şi evoluøia biologicã, executând paşi selectaøi în mod arbitrar. În evoluøia biologicã, speciile eliminã unele din caracteristicile nedorite ale vechii generaøii pe mãsurã ce ele evoluează de la o generaøie la alta. În acest proces, fiecare membru al speciei decide sã reøinã caracteristicile sale în mediul curent sau sã le modifice. Atunci când se utilizează acestã metodã, soluøia curentã a problemei de optimizare este consideratã ca fiind generaøia curentã. Caracteristicile nedorite al soluøiei curente sunt identificate şi eliminate pentru a genera o soluøie mai bunã.

Algoritmii *SE* aparøin clasei euristicilor adaptive, ca şi cãlirea simulatã. Un algoritm adaptiv utilizează un set de parametri de control care sunt modificaøi fie de utilizator, fie de algoritmul însuşi. Astfel, algoritmul se adaptează problemei particulare. Adaptarea parametrilor afectează calitatea soluøiei rezultate. Pentru a se utiliza evoluøia stohasticã, trebuie definitã o funcøie de cost care mãsoarã calitatea soluøiei. Spre deosebire de cãlirea simulatã, funcøia de cost în cazul evoluøiei stohastice poate fi sub

forma unui vector, şi se poate utiliza o relaøie de ordonare pentru a se compara diferiøi vectori de cost.

Parametrii de intrare ai unui algoritm *SE* constau dintr-o soluøie iniøialã S_0, o valoare iniøialã a parametrului de control p_0, şi un parametru *R* care controleazã numãrul de iteraøii. De fiecare datã când se gãseşte o soluøie cu un cost mai mic decât soluøia curentã cea mai bunã, algoritmul decrementeazã contorul prin *R*, recompensându-se astfel prin creşterea numãrului de iteraøii.

3.6.2 Partiționarea prin automate de învățare

Automatele de învãøare au fost utilizate pentru modelarea sistemelor biologice de învãøare şi de asemenea pentru procesul de învãøare a acøiunii optime pe care o oferã un mediu aleator. Învãøarea este realizatã prin interacøiunea cu mediul şi prelucrarea rãspunsurilor sale la acøiunile care au fost alese.

Procesul de învãøare al unui automat poate fi descris astfel: Automatului i se oferã un set de acøiuni de cãtre mediul cu care acesta interacøionează, şi este constrâns sã aleagã una din aceste acøiuni. Atunci când este aleasã o acøiune, automatul este fie recompensat, fie penalizat de cãtre mediu, cu o anumitã probabilitate. Automatul va învãøa alegerea acøiunii optime, care este acøiunea cu probabilitatea minimã de penalizare. În final, automatul va alege aceastã acøiune mai frecvent decât alte acøiuni.

3.6.2.1 Automate de învățare şi partiționarea obiectelor

Automatele stohastice de învãøare pot fi clasificate în douã categorii principale:

1) Automate stohastice cu structurã fixã
2) Automate a cãror structurã evoluează în timp

Un automat stohastic cu structurã fixã (*ASSF*) este un cvintuplu (α, Φ, β, *F*, *G*), unde: $\alpha = \{\alpha_1, \ldots, \alpha_R\}$ este setul acøiunilor din care automatul trebuie sã aleagã, $\Phi = \{\phi_1, \ldots, \phi_S\}$ este setul stãrilor automatului, $\beta = \{0, 1\}$ este setul intrãrilor, F: $\Phi \times \beta \rightarrow \Phi$ este maparea de tranziøie, care defineşte tranziøia stãrii automatului la recepøionarea unei intrãri, iar G: $\Phi \rightarrow \alpha$ este maparea de ieşire, care determinã acøiunea executatã de automat dacã acesta este în starea ϕ_i.

Acøiunea selectatã serveşte ca intrare pentru mediu, care la rândul sãu transmite un rãspuns stohastic $\beta(n)$ la momentul n. $\beta(n)$ este un element din $\beta = \{0, 1\}$ şi este rãspunsul de reacøie al mediului pentru automat. Mediul penalizează ($\beta(n) = 1$) automatul cu penalizarea c_i, care este dependentã de acøiune. Pe baza rãspunsului $\beta(n)$, starea $\phi(n)$ a automatului este actualizatã şi este aleasã o nouã acøiune la momentul n+1.

Problema de partiøionare a grafurilor poate fi rezolvatã în mai multe moduri, considerând aceastã problemã ca una de cãutare sau de exersare bazatã pe parametri. Oommen şi St. Croix [129] au propus o metodã în care problema de partiøionare a grafurilor este rezolvatã prin considerarea acesteia ca o problemã de partiøionare a obiectelor. În acest caz, scopul este de a modela problema astfel încât sã se determine nodurile din *V* care se potrivesc cu alte noduri cu proprietãøi similare. Se încearcã deci impunerea unei mãsuri inteligente de similaritate pentru noduri, astfel încât nodurile similare sã se grupeze. Aceastã mãsurã de similaritate este legatã de apartenenøa sau nu a nodurilor la partiøia optimã.

O altã soluøie a acestei probleme o reprezintã *automatul pentru migrarea obiectelor* (*AMO*), propus de Oommen şi Ma [130]. Acest automat modificã stãrile tuturor celor *W* obiecte, spre deosebire de automatul tradiøional la care obiectele sunt trecute dintr-o stare în alta. Deci, atunci când acest automat este utilizat pentru rezolvarea problemei de echipartiøionare, o soluøie nu este definitã prin starea curentã a *AMO*, ci

prin întreaga structurã a automatului. Funcøia de mapare care defineşte tranziøia automatului dintr-o stare în alta specificã mutarea a douã sau trei obiecte în cadrul acestei structuri. La o interogare $<A_i, A_j>$, dacã A_i şi A_j fac parte din aceeaşi clasã, ambele sunt recompensate şi sunt mutate cu un pas cãtre starea cea mai interioarã asociatã cu acea clasã. Dacã A_i şi A_j fac parte din clase diferite, ele sunt mutate cu un pas cãtre stãrile care sunt învecinate cu stãrile din alte clase, şi atunci când unul din obiecte se aflã într-una din aceste stãri limitã, va migra spre starea corespunzãtoare la o altã acøiune.

3.6.2.2 Automat de învățare pentru partiționarea grafurilor

Automatul de învãøare pentru partiøionarea grafurilor (*APG*) are ca intrãri setul de noduri $V = \{v_1, \ldots, v_{2N}\}$ care trebuie partiøionat în douã subpartiøii de dimensiuni egale, V_1 şi V_2, şi setul de ponderi ale muchiilor. *APG* poate fi generalizat pentru cazul partiøionãrii setului de noduri V în k subpartiøii prin proiectarea acestuia astfel încât sã dispunã de k acøiuni. În continuare se presupune cã automatul trebuie sã rezolve aceastã ultimã problemã de echipartiøionare a setului V în k subseturi.

În timpul procesului de învãøare scopul este de a se colecta noduri similare ale grafului în aceeaşi subpartiøie. Principiul utilizat pentru cuantificarea acestei similaritãøi este analog cu cel utilizat de Kernighan şi Lin. Se considerã cã douã noduri sunt similare dacã ele sunt puternic conectate, şi deci costul muchiei acestora este redus. Explicit, se considerã cã nodurile sunt puternic conectate dacã muchia corespunzãtoare lor are un cost care este de $(1 + \rho)$ ori costul mediu al muchiilor pentru întregul graf, unde ρ este un parametru specificat de utilizator. Similar, nodurile sunt slab conectate dacã muchia lor are un cost care este de $(1 - \rho)$ ori costul mediu al muchiilor [129].

Diferitele stãri din cadrul unei subpartiøii date cuantificã mãsura de certitudine pe care o are sistemul pentru un nod dat aparøinând subpartiøiei respective. La început toate nodurile sunt plasate în subpartiøii alese în mod aleator, în starea limitã (de *Certitudine_Minimã*) a acestor subpartiøii, indicând faptul cã sistemul este nesigur de plasarea corectã a nodurilor. Pe mãsura procesului de învãøare, nodurile similare ale grafului vor fi recompensate pentru apartenenøa lor la aceeaşi subpartiøie, şi vor migra cãtre starea cea mai interioarã a subpartiøiei (de *Certitudine_Maximã*), indicând faptul cã sistemul este mai sigur de plasarea nodurilor în subpartiøiile corecte. În acelaşi timp, alte noduri vor fi penalizate şi vor fi mutate fie spre starea lor limitã, fie în altã subpartiøie, indicând ambiguitatea sistemului în privinøa asocierii lor cu subpartiøia curentã.

Iniøial, automatul are o fazã de preprocesare în care se evalueazã costul mediu al muchiilor. Urmeazã apoi bucla principalã de învãøare a algoritmului. Muchiile sunt procesate în mod repetat, într-o ordine aleatoare. La procesarea unei muchii e_{ij}, dacã v_i şi v_j sunt similare şi aparøin aceleiaşi subpartiøii, nodurile v_i şi v_j sunt recompensate. Acest mod de recompensare se numeşte *Recompensare_Noduri_Similare*. Dacã ele sunt asignate însã unor subpartiøii diferite, automatul este penalizat. Acest mod de penalizare se numeşte *Penalizare_Noduri_Similare*.

Dacã v_i şi v_j sunt nesimilare şi aparøin unor subpartiøii diferite, automatul (şi, în particular, nodurile v_i şi v_j) sunt recompensate. Acest mod de recompensare se numeşte *Recompensare_Noduri_Nesimilare*. Dacã însã nodurile sunt asignate aceleiaşi subpartiøii, automatul este penalizat. Prin analogie cu cele de sus, acest mod de penalizare se numeşte *Penalizare_Noduri_Nesimilare*. Ciclul continuã apoi cu urmãtoarea iteraøie, unde fazele de recompensare şi penalizare se repetã.

3.7 Algoritmi de partiționare propuşi pentru circuite FPGA cu resurse limitate de rutare

3.7.1 Algoritm de partiționare cu echilibrarea numărului de conexiuni

Pentru a elabora o procedurã eficientã de partiøionare, este necesarã o metricã pentru a mãsura congestia canalelor de rutare. Un indicator al acestei congestii este dimensiunea tãieturii, deci numãrul de conexiuni intersectate de o linie de tãieturã din circuit. Aceastã dimensiune reprezintã o limitã inferioarã a numãrului de piste necesare pentru rutarea completã a circuitului.

O linie de tãieturã care trece printr-o zonã cu un numãr mare de conexiuni are de obicei o dimensiune mare a tãieturii. Dacã dimensiunea maximã a tãieturii scade, posibilitatea existenøei unor zone congestionate din punct de vedere al conexiunilor scade în mod corespunzãtor. De aceea, este de dorit minimizarea dimensiunii maxime a tãieturii. Suma dimensiunii tãieturilor pentru toate liniile de tãieturã orizontale şi verticale indicã lungimea totalã a conexiunilor atunci când aceastã lungime pentru o conexiune cu terminale multiple este estimatã prin metoda semi-perimetrului.

Algoritmii de plasare bazaøi pe partiøionarea de tip Kernighan-Lin executã în mod recursiv bipartiøionarea grafului (sau hiper-grafului) care reprezintã circuitul, pânã când graful este suficient de simplu pentru a fi plasat într-o celulã a circuitului. În cazul acestor algoritmi, singura metricã în cadrul funcøiei de cost minimizate este dimensiunea tãieturii. De aceea, se poate obøine o partiøie cu o dimensiune redusã a tãieturii, în care una din porøiuni conøine un numãr mare de conexiuni, în timp ce numãrul conexiunilor din a doua porøiune este redus. Se propune un algoritm de bipartiøionare care øine cont nu numai de dimensiunile celor douã porøiuni, ci şi de distribuøia interconexiunilor din cadrul celor douã porøiuni.

Conexiunile cu terminale multiple se considerã reprezentate printr-un hipergraf. În general, pentru a conecta k terminale, sunt necesare $\max\{k-1, 0\}$ cãi de conectare. Se defineşte *dezechilibrul* unei conexiuni ca fiind diferenøa dintre numãrul cãilor de interconectare necesare pentru conectarea terminalelor din porøiunea din stânga şi numãrul cãilor de interconectare necesare pentru conectarea terminalelor din porøiunea din dreapta. Dezechilibrul unei bipartiøii este definit ca suma valorilor de dezechilibru pentru toate conexiunile. Valoarea absolutã a dezechilibrului unei bipartiøii indicã diferenøa dintre numãrul cãilor de conectare necesare în porøiunea din stânga şi acelaşi numãr din porøiunea din dreapta.

Se utilizează o funcøie de cost care øine cont atât de dimensiunea tãieturii, cât şi de distribuøia numãrului de conexiuni în cadrul unei partiøii

3.7.2 Algoritm genetic pentru partiționare cu echilibrarea numărului de conexiuni

Fie un graf $G = (V, E)$ caracterizat printr-un set de n noduri V şi un set de e muchii E. Algoritmul genetic pentru partiøionare poate fi descris pe scurt astfel. Se porneşte de la o populaøie iniøialã de partiøii. În fiecare generaøie, se obøin partiøii urmaş printr-un proces de încrucişare între douã partiøii pãrinte, cu o ratã specificã de încrucişare. Se utilizează de asemenea un proces de mutaøie pentru a schimba compoziøia structuralã a unui numãr redus de partiøii din populaøie. Din partiøiile iniøiale şi din cele generate prin încrucişare este selectat un set de partiøii care va constitui populaøia generaøiei urmãtoare. Acest proces este continuat pe parcursul mai multor generaøii. În final, partiøia cea mai bunã din cadrul populaøiei este aleasã ca soluøie a problemei de partiøionare.

Fiecare soluøie a problemei este reprezentatã printr-un cromozom, care este un şir binar. Un cromozom corespunde deci unei bisecøii a grafului. Numãrul de gene ale unui cromozom este egal cu *n*, numãrul de vârfuri ale grafului. O genã are valoarea 0 dacã vârful corespunzãtor se aflã în partea stângã a bisecøiei, şi are valoarea 1 dacã vârful se aflã în partea dreaptã a bisecøiei.

Pentru *iniøializare*, algoritmul creazã în mod aleator N_p soluøii, unde N_p este dimensiunea populaøiei. Presupunând cã numãrul de vârfuri ale grafului este par, singura restricøie asupra unui cromozom este cã acesta trebuie sã conøinã un numãr egal de valori 0 şi 1.

Fiecãrei soluøii din cadrul populaøiei i se asignează o valoare de viabilitate calculatã din dimensiunea tãieturii. Valoarea viabilitãøii F_i a soluøiei *i* se calculează astfel:

$$F_i = 1 / (Dim\_tăietură + PONDERE \times Dezechilibru) \quad (3.48)$$

unde *Dim_tãieturã* este dimensiunea tãieturii soluøiei, *PONDERE* este o constantã, iar *Dezechilibru* este diferenøa între numãrul de conexiuni din porøiunea din stânga a partiøiei şi numãrul de conexiuni din porøiunea din dreapta a partiøiei. Un cromozom este selectat ca un pãrinte cu o probabilitate care este proporøionalã cu valoarea sa de viabilitate.

Un *operator de încrucişare* creazã un nou cromozom urmaş prin combinarea unor pãrøi ale celor doi cromozomi pãrinte. Cel mai simplu operator de încrucişare selectează în mod aleator un punct de tãieturã, care este acelaşi pentru ambii cromozomi pãrinte. Acest punct divide cromozomul în douã pãrøi, partea stângã şi partea dreaptã. Partea stângã a pãrintelui 1 şi partea dreaptã a pãrintelui 2 sunt copiate în aceleaşi poziøii ale cromozomului urmaş. Fie acesta urmaşul 1.

Algoritmul propus utilizează şi un alt operator de încrucişare, care este acelaşi cu cel descris, cu excepøia faptului cã se copiază valorile complementare ale pãrøii drepte a pãrintelui 2, în timp ce partea stângã a pãrintelui 1 este copiatã nemodificat. Fie acesta urmaşul 2. Algoritmul selectează cel mai bun dintre cei doi urmaşi. Motivul pentru utilizarea a doi operatori de încrucişare este urmãtorul. Dacã doi cromozomi sunt exact (sau aproape exact) complementul unuia faøã de celãlalt, aceştia reprezintã aceeaşi bisecøie. În acest caz, primul operator va crea o inconsistenøã într-un cromozom urmaş, şi în consecinøã acest cromozom va avea o calitate redusã.

Operatorul de mutaøie utilizat funcøionează astfel: sunt selectate în mod aleator *m* poziøii în cadrul cromozomului, şi valorile lor sunt inversate. Valoarea lui *m* este un întreg în intervalul [0, *n*/10]. Dupã operaøiile de încrucişare şi mutaøie, un urmaş poate avea un numãr diferit de valori 0 şi 1. Algoritmul calculează diferenøa dintre numãrul valorilor de 1 şi 0. Apoi selectează un punct aleator din cromozom şi modificã numãrul necesar de valori de 1 în 0 (sau 0 în 1) începând din acel punct spre dreapta (continuând de la stânga dacã este necesar).

Dupã generarea unui nou urmaş, algoritmul înlocuieşte un membru al populaøiei cu acest urmaş. Calitatea soluøiilor depinde în mare mãsurã de *metoda de înlocuire*. Trebuie aleasã o metodã de înlocuire care generează soluøii de calitate într-un timp rezonabil. În algoritmul de partiøionare propus, se combinã metoda de înlocuire din [32] cu cea de tip *Genitor*. Se încearcã mai întâi înlocuirea pãrintelui cel mai similar, pe baza distanøei Hamming; dacã urmaşul este de calitate mai slabã decât ambii pãrinøi, se înlocuieşte membrul cel mai inferior calitativ al populaøiei. Scopul este de a se menøine diversitatea populaøiei, fãrã a se creşte în mod semnificativ timpul consumat inutil. Rezultatele obøinute prin aceastã metodã combinatã sunt superioare celor obøinute prin metoda de tip *Genitor*.

Criteriul de terminare pentru un mare numãr de algoritmi genetici este execuøia pentru un numãr fix de generaøii. Un criteriu mai avantajos este terminarea algoritmului atunci când diversitatea populaøiei scade sub un anumit prag. Algoritmul se terminã atunci când 80% a populaøiei este ocupatã cu soluøii de aceeaşi calitate. Numãrul maxim de iteraøii este limitat la 2000.

3.7.3 Rezultate experimentale

Algoritmul de partiøionare propus pentru echilibrarea numãrului de conexiuni a fost implementat în limbajul C. Experimentele au fost efectuate pe un calculator IBM PC cu un procesor Pentium de 133 MHz, sub sistemul de operare Windows NT Version 4.0. S-a utilizat un numãr de nouã circuite de test din cadrul setului de circuite al centrului *MCNC* (*Microelectronics Center of North Carolina*). Lista de conexiuni a circuitelor de test a fost convertitã din formatul *EDIF* (*Electronic Design Interchange Format*) în formatul listei de conexiuni utilizate de programul de partiøionare. Algoritmul de partiøionare a fost aplicat asupra listelor de conexiuni obøinute astfel.

În cadrul experimentelor efectuate s-a urmãrit care este valoarea tãieturii obøinute în urma bipartiøionãrii fiecãrui circuit în douã cazuri: atunci când nu se urmãreşte echilibrarea numãrului de conexiuni din cele douã partiøii, şi atunci când se realizează şi echilibrarea numãrului de conexiuni. În primul caz se obøine o anumitã creştere a dimensiunii tãieturii faøã de cazul al doilea, ceea ce este explicabil, deoarece în primul caz singura metricã utilizatã este cea care indicã dimensiunea tãieturii.

În cazul bipartiøionãrii, aceastã dimensiune este cea a tãieturii din centrul circuitului. Pentru a mãsura dimensiuea tãieturii şi în alte zone ale circuitului, algoritmul de bipartiøionare a fost aplicat recursiv pânã la obøinerea unor zone care conøin o singurã celulã, urmãrindu-se care este suma tãieturilor în cele douã cazuri. S-a observat cã pentru şase din cele opt circuite utilizate pentru experimentãri s-a obøinut o reducere a sumei tãieturilor, pe lângã echilibrarea numãrului de conexiuni. Reducerea medie pentru circuitele de test utilizate este de 8.5 %.

Algoritmul genetic pentru partiøionare a fost comparat cu un algoritm de partiøionare prin metoda cãlirii simulate. Experimentele au arãtat cã timpul de execuøie al algoritmului genetic este mai redus, rezultatele obøinute fiind comparabile cu cele obøinute prin metoda cãlirii simulate.

3.8 Concluzii

În acest capitol a fost prezentatã o problemã importantã care apare în cadrul proiectãrii fizice, şi anume partiøionarea circuitelor. Aceastã problemã a fost prezentatã atât ca o etapã de proiectare pentru divizarea unui sistem în mai multe pãrøi care pot fi implementate prin componente separate, cât şi ca o metodã algoritmicã pentru rezolvarea problemelor complexe de optimizare care apar în sinteza logicã sau în proiectarea fizicã a circuitelor VLSI în general şi FPGA în particular.

Problema de bipartiøionare în care cele douã partiøii au dimensiuni egale a fost examinatã mai detaliat, datoritã importanøei sale practice. Aceastã partiøionare este utilizatã în cadrul procedurii de plasare pe baza tãieturii minime. Algoritmul Kernighan-Lin este unul din cele mai utilizate pentru rezolvarea problemei de bipartiøionare. Este un algoritm cu îmbunãtãøire iterativã, care poate fi extins şi pentru rezolvarea unor probleme mai generale de partiøionare. O extindere a algoritmului Kernighan-Lin şi o implementare mai eficientã a acestuia a fost realizatã de Fiduccia şi Mattheyses, euristica acestora luând în considerare atât conexiunile multipin, cât şi dimensiunile elementelor de circuit.

Cãlirea simulatã este o metodã stohasticã de îmbunãtãøire iterativã utilizatã pentru rezolvarea diferitelor probleme de optimizare, inclusiv pentru cea de partiøionare. Prin metoda de cãlire simulatã se obøin anumite avantaje în privinøa calitãøii soluøiei, dar timpul de calcul consumat poate fi foarte ridicat. Partiøionarea prin metoda tãieturii proporøionale se bazează pe o metricã propusã de Wei şi Cheng. Cheng şi Wei au propus o metodã de bipartiøionare cu performanøe stabile, care nu necesitã generarea unui mare numãr de configuraøii iniøiale. Metodele spectrale utilizează vectori proprii şi valori proprii ale matricii de adiacenøã a grafului care descrie circuitul.

Metodele bazate pe fluxul în reøele utilizează fluxul direcøionat al semnalelor pentru îmbunătăøirea performanøelor sistemului.

A fost descrisă de asemenea o metodă probabilistică de partiøionare, care poate determina implicaøiile viitoare ale mutării unui nod în orice etapă a procesului de partiøionare. Au fost descrise şi unele metode neconvenøionale de partiøionare: partiøionarea prin evoluøie stohastică şi cea prin automate de învăøare. Metoda de evoluøie stohastică descrisă presupune că nodurile circuitului au dimensiuni diferite. Automatul de învăøare descris este numit automat pentru migrarea obiectelor. Acest automat modifică stările tuturor obiectelor, spre deosebire de automatul tradiøional la care obiectele sunt trecute dintr-o stare în alta.

În cadrul capitolului s-au propus doi algoritmi de bipartiøionare pentru circuitele FPGA cu resurse limitate de rutare. Primul algoritm se bazează pe metoda tăieturii minime, şi urmăreşte echilibrarea numărului de conexiuni din cadrul partiøiilor, minimizând în acelaşi timp lungimea totală a interconexiunilor. A fost propusă o metrică mai adecvată pentru partiøionarea utilizată la plasarea circuitelor FPGA la care principalul obiectiv este asigurarea rutabilităøii. Experimentele efectuate arată că prin aplicarea acestui algoritm se obøine o reducere a dimensiunii tăieturii.

Al doilea algoritm propus este un algoritm genetic pentru partiøionare, cu un obiectiv similar cu primul algoritm. Algoritmul utilizează doi operatori de încrucişare în loc de unul singur, al doilea operator fiind prevăzut pentru situaøia în care doi cromozomi sunt exact complementul unuia faøă de celălalt, şi deci aceştia reprezintă aceeaşi bisecøie. Metoda de înlocuire utilizată este diferită de cea a algoritmilor tradiøionali, având ca scop principal menøinerea diversităøii populaøiei. Criteriul de terminare este de asemenea diferit, algoritmul fiind terminat atunci când diversitatea populaøiei scade sub un anumit prag.

4. PLASAREA MODULELOR CU OBIECTIVUL ASIGURĂRII RUTABILITĂŢII CIRCUITELOR

4.1 Introducere

Plasarea este procesul de aranjare a componentelor unui circuit pe o suprafaøă. În cazul circuitelor FPGA, plasarea semnifică asignarea funcøiilor logice diferitelor celule ale circuitului, a căror poziøie este fixă. Plasarea este o etapă importantă a procesului de proiectare, deoarece în această etapă se iau cele mai importante decizii. De exemplu, plasarea celulelor standard ale circuitelor VLSI are un impact major asupra vitezei şi costului final al circuitului.

Lungimea totală a conexiunilor ω reprezintă o metrică utilizată pe scară largă pentru aprecierea calităøii plasării [146]. O plasare care necesită un spaøiu mare pentru conexiuni va necesita conexiuni de lungime mare, şi deci lungimea totală a conexiunilor va avea o valoare mare. Deci, lungimea totală a conexiunilor ω este o metrică potrivită pentru suprafaøa ocupată de circuit.

4.2 Definirea problemei de plasare

Problema de plasare poate fi definită astfel. Fiind dat un set de module $M = \{m_1, m_2, \ldots, m_n\}$ şi un set de semnale $S = \{s_1, s_2, \ldots, s_k\}$, se asociază cu fiecare modul $m_i \in M$ un set de semnale S_{m_i}, unde $S_{m_i} \subseteq S$. Similar, cu fiecare semnal $s_i \in S$ se aso-

ciazã un set de module M_{s_i}, unde $M_{s_i} = \{m_j \mid s_i \in S_{m_j}\}$. M_{s_i} se numešte o conexiune de semnal. Este dat de asemenea şi un set de locaøii $L = \{L_1, L_2, \ldots, L_p\}$, unde $p \geq n$. Problema de plasare este de a asigna fiecare modul $m_i \in M$ unei locaøii unice L_j astfel încât sã fie optimizat un anumit obiectiv.

4.3 Funcții de cost şi restricții

În cadrul procesului de proiectare, etapa de *plasare* este urmatã de cea de *rutare*. O plasare este acceptabilã dacã se poate obøine o rutare de 100% în cadrul suprafeøei date. Funcøia obiectiv care trebuie minimizatã se poate scrie ca o sumã dintre γ_1 şi γ_2. În cele mai multe cazuri, γ_1 este lungimea totalã estimatã a conexiunilor. În general, γ_2 reprezintã penalizãri pentru soluøiile non-fezabile, fiind costul violãrii restricøiilor.

Executarea efectivã a rutãrii pentru a compara diferitele soluøii de plasare nu este practicã. De aceea, se utilizeazã diferite *estimãri* ale lungimii conexiunilor.

4.3.1 Estimarea lungimii conexiunilor

Viteza estimãrii are o influenøã fundamentalã asupra performanøelor algoritmului de plasare. De aceea, procedura de estimare trebuie sã fie cât mai rapidã. În plus, eroarea de estimare trebuie sã fie aceeaşi pentru toate conexiunile.

O presupunere realisticã pentru estimarea lungimii totale a conexiunilor este cã rutarea utilizeazã geometria Manhattan [146]. Pentru o conexiune cu doi pini între modulul *i* şi modulul *j*, lungimea Manhattan este $r_{ij} + c_{ij}$, unde r_{ij} şi c_{ij} reprezintã numãrul de linii şi coloane care separã locaøiile celor douã module. Totuşi, nu toate conexiunile sunt cu doi pini. Este necesarã deci o metodã pentru a estima lungimea unei conexiuni multipunct. Existã diferite tehnici utilizate, fiecare din acestea având avantaje şi dezavantaje.

Metoda semi-perimetrului. Aceasta este o aproximare eficientã şi foarte utilizatã pentru a estima lungimea unei conexiuni. Metoda constã în determinarea celui mai mic dreptunghi care cuprinde toøi pinii conexiunii respective. Lungimea estimatã a interconexiunilor este jumãtatea perimetrului acestui dreptunghi.

Metoda aproximãrii arborelui Steiner. Un *arbore Steiner* reprezintã calea cea mai scurtã pentru conectarea unui set de pini. În aceastã metodã, pot exista ramificãri din orice punct al unei conexiuni pentru conectarea la aløi pini. Existã diferite metode pentru determinarea unei aproximãri a arborelui Steiner.

Metoda arborelui de acoperire minim. Spre deosebire de arborele Steiner, într-un arbore de acoperire minim ramificarea este permisã numai în poziøiile pinilor. Pentru o conexiune cu *n* pini, arborele poate fi construit prin determinarea distanøelor dintre toate perechile posibile de pini, şi conectarea celor mai mici (*n* - 1) muchii care nu formeazã cicluri.

4.3.2 Minimizarea lungimii totale a conexiunilor

O funcøie obiectiv a cãrei minimizare se urmãreşte şi care este des utilizatã este $L(P)$, lungimea totalã ponderatã pentru toate conexiunile de semnal, exprimatã ca:

$$L(P) = \sum_{n \in N} w_n \cdot d_n \tag{4.3}$$

unde d_n este lungimea estimatã a conexiunii *n*, w_n este ponderea conexiunii *n*, iar *N* este setul de conexiuni.

4.3.3 Minimizarea tăieturii maxime

Considerăm un spaţiu pentru plasare sub forma unei suprafeţe rectangulare, în care a fost plasat un circuit. Linia verticală de la $x = x_i$ divide suprafaţa într-o regiune din stânga L_i şi o regiune din dreapta R_i. Faţă de această linie de tăietură, conexiunile se pot clasifica astfel:

a) Conexiuni care se află în totalitate la stânga liniei de tăietură. Toţi pinii unor asemenea conexiuni se vor afla în L_i.

b) Conexiuni care se află în totalitate la dreapta liniei de tăietură. Toţi pinii unor asemenea conexiuni se vor afla în R_i.

c) Conexiuni care sunt tăiate de linie. Fiecare conexiune din această clasă va avea în mod necesar cel puţin un pin în L_i şi cel puţin un pin în R_i.

Fie $\Phi_P(x_i)$ numărul de conexiuni de tipul c) pentru plasarea P tăiată de linia x_i. $\Phi_P(x_i)$ este o funcţie de plasarea P. Pentru o anumită plasare P, fie $X(P)$ valoarea maximă a $\Phi_P(x_i)$ pentru fiecare i, deci:

$$X(P) = \max_i[\Phi_P(x_i)] \tag{4.4}$$

În mod similar, se pot defini linii de tăietură orizontale y_j şi tăietura verticală maximă $Y(P)$ ca fiind:

$$Y(P) = \max_j[\Phi_P(y_j)] \tag{4.5}$$

Reducerea tăieturii orizontale $X(P)$ şi a celei verticale $Y(P)$ prin selectarea unei plasări P corespunzătoare poate mări probabilitatea rutării circuitului. În plus, minimizarea $X(P)$ şi $Y(P)$ poate avea de asemenea o influenţă benefică asupra lungimii totale a conexiunilor $L(P)$.

4.4 Sinteza metodelor de plasare

Metodele euristice pentru plasare se pot clasifica în două categorii: *constructive* şi *iterative*. Aceste metode sunt de două tipuri: metode bazate pe partiţionare şi metode analitice. În ultimul timp, au fost obţinute soluţii de calitate prin combinarea ambelor strategii [67].

În literatură a fost publicat un număr mare de algoritmi pentru plasare. Metodele bazate pe partiţionare implică aplicarea recursivă a unui algoritm de partiţionare, de obicei algoritmul Kernighan-Lin sau algoritmul de călire simulată [102]. Deşi metoda de călire simulată poate obţine soluţia optimă globală, timpul de calcul necesar pentru circuite de dimensiuni mari este foarte mare, şi de multe ori nu este acceptabil în practică. Pentru a elimina dezavantajul complexităţii ridicate a algoritmului de călire simulată, au fost propuse mai multe tehnici de creştere a vitezei pentru acest algoritm [107], [119].

Hamada *et al.* [88] au propus o tehnică de plasare pe baza partiţionării ierarhice prin algoritmul de tăietură proporţională elaborat de Cheng şi Wei [50]. Prin aplicarea recursivă a partiţionării, circuitul este descompus într-un arbore de grupuri, iar problema iniţială de plasare este soluţionată printr-o secvenţă de procese de călire simulată, unde fiecare grup este tratat ca o super-componentă. Această metodă reduce în mod considerabil complexitatea problemei.

Pentru soluţionarea problemei de plasare prin tehnici iterative a fost propusă aplicarea metodei de asignare liniară, o metodă exactă şi eficientă din punct de vedere computaţional [67]. Această metodă a fost utilizată pentru translatarea unei plasări globale conţinând celule suprapuse într-o plasare finală prin minimizarea distanţei cu care celulele sunt mutate din poziţiile lor suprapuse. Algoritmii de asignare liniară au

fost propuši de asemenea pentru soluøionarea problemei speciale de plasare în care toate celulele au aceeaši dimensiune şi sunt specificate locaøiile posibile ale acestora.

Pentru rezolvarea problemei de plasare liniarã, au fost propuse diferite metode. Li *et al.* [109] au propus o metodã spectralã în care se utilizeazã o funcøie obiectiv rezultatã dintr-o combinaøie a unei funcøii liniare cu o funcøie cuadraticã. Se utilizeazã astfel atât avantajul unei funcøii obiectiv liniare, care permite obøinerea unei plasãri cu o lungime mai redusã a conexiunilor, cât şi avantajul unei funcøii cuadratice, care tinde sã plaseze componentele mai dispersat, rezultând o soluøie mai fezabilã.

Problema de plasare poate fi transformatã într-o problemã de optimizare numericã. Hanan şi Kurtzberg au redus problema de plasare la cea de rezolvare a unui set de ecuaøii liniare pentru a determina locaøiile de echilibru ale celulelor [146].

Plasarea orientatã pe performanøe a fost studiatã de numeroši autori, în special plasarea cu restricøii de timp. Metodele raportate pot fi clasificate în trei categorii [146]. Prima categorie transformã restricøiile de timp de pe cãile critice în ponderi ale conexiunilor. A doua categorie transformã restricøiile de timp ale cãilor în limite de timp ale conexiunilor. Aceste limite de timp sunt convertite în limite de lungime ale conexiunilor şi sunt furnizate programului de plasare. A treia metodã constã în furnizarea pentru programul de plasare a unui set de cãi critice, împreunã cu cerinøele lor de timp. Aceste cãi sunt monitorizate în timpul procesului de plasare.

Problema de plasare a circuitelor FPGA a fost abordatã prin diferite metode, dintre care şi prin metoda cãlirii simulate, într-un mod similar cu plasarea celulelor standard. În timp ce tehnicile elaborate pentru celulele standard sunt suficiente pentru acele circuite FPGA la care o mare porøiune a spaøiului de pe cip este dedicatã resurselor de rutare [178], în cazul arhitecturilor FPGA cu resurse limitate de rutare sunt necesare tehnici speciale. Ebeling *et al.* [71] au elaborat un program de plasare pentru circuitele FPGA Triptych, program care se bazeazã pe metoda de cãlire simulatã. Beetem [18] a propus un algoritm de îmbunãtãøire iterativã pentru plasarea şi rutarea simultanã a circuitelor FPGA. Nag şi Roy [125] a prezentat un algoritm incremental de plasare pentru circuitele FPGA cu o arhitecturã bazatã pe rânduri de celule, care analizeazã informaøiile de întârziere a semnalelor pentru a obøine plasãri de calitate mai bunã. Togawa *et al.* [167] au propus o metodã pentru plasarea şi rutarea simultanã a circuitelor FPGA simetrice, metodã bazatã pe bipartiøionarea ierarhicã.

Gao [82] a elaborat algoritmi de plasare orientaøi pe performanøe bazaøi pe conexiuni şi pe cãi de rutare, pentru reøele de porøi, macro-celule şi circuite FPGA Xilinx, cu scopul minimizãrii întârzierii semnalelor la pinii de ieşire şi a nesimetriei semnalelor la intrãrile modulelor. În cazul algoritmului bazat pe conexiuni, cerinøele de întârziere sunt translatate mai întâi în constrângeri de proiectare fizicã. Algoritmul de plasare generează apoi o plasare ghidatã de aceste constrângeri. În cazul algoritmului bazat pe cãi de rutare, întârzierile cãilor sunt considerate în mod explicit în timpul procesului de plasare. Acest algoritm încearcã sã minimizeze lungimea totalã a conexiunilor şi timpii de întârziere la pinii de ieşire.

Pentru rezolvarea problemei de plasare au fost propuse diferite metode neconvenøionale, ca reøelele neuronale, algoritmii genetici, logica fuzzy, sau prelucrarea paralelã. Yu [187] a modificat reøelele neuronale introduse de Hopfield şi Tank pentru rezolvarea problemei de plasare. Funcøia de energie utilizatã de Hopfield are mai multe minime, dintre care unele sunt minime locale; reøeaua neuronalã poate converge în oricare din acestea. În plus, este dificilã determinarea parametrilor reøelei. Rezultatele obøinute de Yu nu au fost satisfãcãtoare, din cauza timpilor mari de simulare şi dependenøa soluøiilor de parametrii reøelei.

Algoritmii genetici au fost de asemenea utilizaøi pentru plasarea celulelor circuitelor VLSI [124], [152]. Mohan şi Mazumder [124] au elaborat un algoritm genetic pentru plasarea celulelor standard. A fost implementatã atât o versiune serialã a algoritmului, cât şi una paralelã, care rulează pe o reøea de staøii de lucru. Creşterea de vitezã obøinutã este liniarã cu numãrul de procesoare utilizate.

Plasarea este o problemă cu obiective multiple, unde numeroase decizii care sunt luate în timpul căutării soluţiei sunt calitative. O abordare potrivită a acestei categorii de probleme este utilizarea logicii fuzzy. O descriere a problemei de plasare care se bazează pe logica fuzzy a fost publicată de Lin şi Shragowitz [113].

Au fost realizate diferite implementări paralele ale metodei de călire simulată pe diferite tipuri de calculatoare paralele: cu memorie partajată, cu transmitere de mesaje şi memorie locală, şi calculatoare cu paralelism masiv, ca de exemplu Connection Machine. Kravitz şi Rutenbar [104] au prezentat un algoritm de călire simulată pentru celule standard, care a fost implementat pe un multiprocesor. Aceştia au arătat că algoritmul de călire simulată poate fi accelerat în două moduri pe un multiprocesor cu memorie partajată: prin executarea mai multor mutări în paralel, şi prin executarea în paralel a prelucrărilor necesare pentru fiecare mutare.

Rose *et al.* [141] au conceput euristici pentru plasarea paralelă a celulelor standard cu o calitate echivalentă cu cea a călirii simulate. Aceştia au utilizat o metodă rapidă bazată pe tăietura minimă pentru a evita partea de călire lentă la temperaturi înalte. În faza de călire la temperaturi joase, spaţiul din cadrul circuitului este partiţionat şi este asignat diferitelor procesoare, astfel încât fiecare procesor mută celule într-o anumită zonă şi, ori de câte ori o mutare este acceptată, transmite rezultatul tuturor procesoarelor.

4.4.1 Plasarea constructivă inițială

Un algoritm constructiv generează o configuraţie de plasare completă numai la sfârşitul întregului proces. Un asemenea algoritm se utilizează adesea pentru generarea unei plasări iniţiale, urmând ca aceasta să fie îmbunătăţită printr-o metodă iterativă. Plasarea iniţială este esenţială pentru obţinerea unei soluţii optime. Există diferite soluţii optime, corespunzătoare diferitelor plasări iniţiale.

Se pot utiliza diferite euristici pentru deciziile care trebuie luate. De exemplu, o euristică posibilă pentru selecţie este alegerea acelui modul care este cel mai puternic conectat cu plasarea parţială existentă. Presupunând că plasarea parţială este formată din modulele m_1, m_2, …, m_i, se examinează fiecare din modulele neplasate m_j şi se calculează cantitatea

$$A_{mj} = \sum_{k=1}^{i} c_{m_j m_k} \tag{4.9}$$

unde $c_{m_j m_k}$ reprezintă conectivitatea între modulul neplasat m_j şi un modul plasat m_k. Astfel, A_{mj} indică numărul de conexiuni de la m_j la modulele deja plasate {m_1, m_2, …, m_i }. Se va selecta modulul pentru care A_{mj} este maxim. Această strategie este cunoscută sub numele de strategie de *conectivitate maximă*.

4.4.2 Plasarea pe baza tăieturii minime

În secţiunea 4.3 au fost prezentate trei funcţii obiectiv, şi anume $X(P)$, $Y(P)$ şi $L(P)$. S-a arătat că prin minimizarea $X(P)$, tăietura orizontală maximă, şi prin minimizarea $Y(P)$, tăietura verticală maximă, se va îmbunătăţi rutabilitatea unei plasări pentru o reţea de porţi. Minimizarea funcţiei $X(P)$ este strâns legată de problema de bipartiţionare. De aceea, se aplică un algoritm de partiţionare pentru circuitul dat pentru a genera două blocuri A şi B, se plasează modulele din blocul A la stânga unei linii imaginare de tăietură verticală c_1, şi se plasează modulele din blocul B la dreapta liniei de tăietură c_1. Setul de tăietură obţinut de algoritm este numărul de conexiuni orizontale tăiate de c_1, şi este notat cu $\Phi_P(c_1)$.

Presupunem că se repetă procesul pentru blocurile A şi B, deci se consideră blocul A ca un circuit şi se partiţionează în două blocuri A_1 şi A_2, utilizând o linie de

tãieturã verticalã c_2. Similar, blocul *B* se partiøioneazã în douã blocuri B_1 øi B_2, utilizând o linie de tãieturã verticalã c_3 (Figura 4.1). Acest proces poate fi repetat prin introducerea altor linii de tãieturã. Presupunem cã procedura de partiøionare utilizatã generează o partiøie optimã. Pentru întregul circuit, $\Phi_P(c_1)$ este tãietura minimã posibilã. Similar, $\Phi_P(c_2)$ este tãietura minimã posibilã pentru subcircuitul *A*.

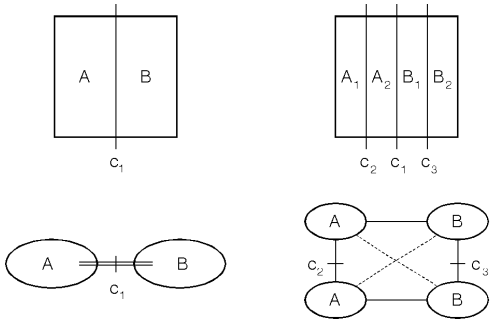

Figura 4.1. Utilizarea partiøionãrii pentru a reduce *X*(*P*).

Minimizarea funcøiilor *X*(*P*), *Y*(*P*) sau *L*(*P*) este foarte dificilã din punct de vedere computaøional. Pentru simplificarea problemei, se utilizează o *funcøie obiectiv secvenøialã*, notatã cu *F*(*P*), a cãrei valoare apropiatã de cea minimã este mai uøor de obøinut.

$$F(P) = \min\left[\Phi_P(c_r)\right] \mid \min\left[\Phi_P(c_{r\text{-}1})\right] \mid \ldots \mid \min\left[\Phi_P(c_1)\right] \qquad (4.32)$$

unde c_1, c_2, …, c_r este o secvenøã ordonatã de linii de tãieturã verticale sau orizontale.

Algoritmul de plasare pe baza tãieturii minime presupune cã este disponibilã o secvenøã ordonatã de *r* linii de tãieturã. Aceste *r* linii de tãieturã împart suprafaøa de plasare în locaøii.

4.4.3 Plasarea prin metoda călirii simulate

4.4.3.1 Aplicarea algoritmului de călire simulată pentru plasare

Algoritmul de cãlire simulatã poate fi modificat pentru plasarea celulelor prin alegerea unei funcøii adecvate de *perturbaøie* pentru a genera o nouã configuraøie de plasare, øi prin definirea unei funcøii corespunzãtoare de *acceptare*. O funcøie simplã de vecinãtate se obøine prin interschimbarea perechilor, în care sunt alese douã locaøii øi se interschimã conøinutul lor. Alte metode pentru generarea unor stãri de vecinãtate sunt mutarea unei celule selectate arbitrar într-o locaøie arbitrarã, rotirea celulelor dacã aceasta este permisã de strategia de amplasare, sau orice altã mutare care poate modifica lungimea conexiunilor.

Fie Δh = (*Cost*(*NewS*) - *Cost*(*S*)) modificarea lungimii estimate a conexiunilor datoritã unei interschimbãri, unde *Cost*(*S*) este vechea lungime a conexiunilor, iar *Cost*(*NewS*) este lungimea dupã perturbaøie. Interschimbarea este acceptatã dacã $\Delta h < 0$ (deci, *Cost*(*NewS*) < *Cost*(*S*)) sau dacã funcøia de acceptare (*random* < $e^{-\Delta h / T}$) este adevãratã, unde *random* este un numãr aleator între 0 øi 1, generat în mod uniform, iar *T* este valoarea curentã a temperaturii.

4.4.3.2 Algoritmul TimberWolf

Pachetul de programe TimberWolf3.2 [151] este destinat configuraøiilor de circuite cu celule standard. Pe baza datelor de intrare øi a parametrilor furnizaøi de utili-

zator, programul de plasare construieşte o topologie de circuit cu celule standard. Acešti parametri, împreunã cu lãøimea totalã a celulelor care trebuie plasate, permite programului sã calculeze poziøia iniøialã şi lungimile rândurilor orizontale. Blocurile de macrouri sunt plasate urmãtoarele, urmate de plasarea celulelor de I/E. Blocurile de macrouri şi celulele de I/E îşi pãstreazã poziøia iniøialã, fiind optimizatã numai plasarea celulelor standard.

După plasarea iniøialã, algoritmul executã plasarea şi rutarea în trei etape distincte. În prima etapã, celulele sunt plasate astfel încât sã se minimizeze lungimea estimatã a conexiunilor. În a doua etapã, sunt inserate celule de trecere după cum este necesar, lungimea conexiunilor este minimizatã din nou, şi se executã rutarea globalã preliminarã. În a treia etapã, sunt efectuate modificări locale ale plasării pentru a reduce numărul pistelor de rutare necesare. Se va prezenta în continuare prima etapă a algoritmului, care utilizează călirea simulatã pentru plasare.

4.4.4 Plasarea prin partiționare ierarhică

Deşi metoda călirii simulate a fost aplicatã cu succes pentru plasarea circuitelor, pe măsura creşterii dimensiunii circuitului timpul de calcul necesar devine inacceptabil în practicã. Pentru reducerea timpului de calcul necesar algoritmului de călire simulatã au fost utilizate diferite tehnici [88] [107] [119]. Hamada *et al.* [88] au propus utilizarea metodei de partiøionare ierarhicã pe baza tãieturii proporøionale, elaboratã de Cheng şi Wei [50], urmatã de aplicarea călirii simulate multinivel.

În prima etapã, circuitul este descompus într-un arbore de grupuri prin aplicarea recursivã a metodei de partiøionare pe baza tãieturii proporøionale. Se reduce astfel în mod semnificativ complexitatea problemei. Rezultatul partiøionării este reprezentat printr-un arbore binar, a cărui rãdãcinã reprezintã întregul circuit. În etapa a doua, fiecare grup este considerat ca o super-componentã, şi se aplicã metoda de călire simulatã pentru acestea. În etapa a treia, se integrează soluøiile subproblemelor prin utilizarea metodei ferestrelor mobile. În fiecare pas, este definitã o fereastrã peste configuraøia curentã de plasare. Grupurile de sub aceastã fereastrã sunt repartiøionate, şi se aplicã algoritmul de călire simulatã noilor grupuri generate. Fereastra este apoi deplasatã şi ciclul continuã.

4.4.5 Plasarea prin metode numerice

Problema de plasare poate fi transformatã într-o problemã de optimizare numericã. În această secøiune se va prezenta o metodã numitã plasare controlatã de forøe, elaboratã de Hanan şi Kurtzberg [146]. Problema de plasare este redusã la soluøionarea unui sistem de ecuaøii liniare pentru a determina locaøiile de echilibru ale celulelor.

Ideea de bazã a acestei metode este cã celulele interconectate exercitã forøe unele asupra altora. Mărimea forøei F exercitate de o celulã i asupra altei celule j este proporøionalã cu distanøa care le separã. Aceasta este o analogie cu legea lui Hooke din mecanicã, care se referã la forøele exercitate între douã mase conectate printr-un arc. Dacã masele se aflã la o distanøã d şi constanta arcului este k, forøa cu care masele se atrag este $k \times d$. Presupunem cã o celulã a este conectatã cu o altã celulã b printr-o conexiune cu ponderea w_{ab}. Fie d_{ab} distanøa între a şi b. Forøa de atracøie dintre celule este proporøionalã cu produsul $w_{ab} \times d_{ab}$. O celulã i conectatã cu mai multe celule j aflate la distanøe d_{ij} prin conexiuni cu ponderi w_{ij} este atrasã cu o forøã totalã F_i:

$$F_i = \sum_j w_{ij} \cdot d_{ij} \tag{4.41}$$

Dacã celula i dintr-un asemenea sistem îşi poate modifica poziøia, această deplasare se va realiza în direcøia forøei F_i până când forøa rezultantã asupra celulei va fi zero. Locaøia în care se va deplasa celula este numitã locaøie destinaøie de forøã zero.

În ecuaøia (4.41), F_i reprezintã lungimea totalã ponderatã a conexiunilor care pornesc de la celula *i*. Atunci când toate celulele se deplasează în locaøiile lor de forøã zero, suma pãtratelor distanøelor este minimizatã.

4.4.6 Plasarea liniară prin metode spectrale

Plasarea liniarã este o problemã fundamentalã pentru proiectarea circuitelor VLSI. O plasare liniarã de calitate mai bunã are ca efect reducerea lungimii conexiunilor. Au fost propuse diferite metode care utilizează vectori proprii, atât pentru problema de partiøionare, cât şi pentru plasarea liniarã [39] [40] [86] [109].

În literaturã s-au efectuat comparaøii între funcøiile obiectiv liniare şi cuadratice. S-a constatat cã prin utilizarea unei funcøii liniare se obøine o plasare de calitate mai bunã din punct de vedere a lungimii conexiunilor. În [139], utilizarea unei funcøii liniare pentru plasare a permis de asemenea obøinerea unor îmbunãtãøiri importante ale partiøionãrii din punct de vedere a capacitãøii tãieturii. Utilizarea unei funcøii obiectiv cuadratice are însã ca efect obøinerea unui numãr mai redus de conexiuni foarte lungi faøã de cazul unei funcøii obiectiv liniare. În cazul funcøiei cuadratice, deviaøia standard a lungimii conexiunilor este mai micã decât în cazul funcøiei liniare [109]. Aceasta înseamnã cã funcøia cuadraticã are tendinøa sã plaseze componentele într-un mod mai dispersat, rezultând mai puøine componente suprapuse. Dezavantajul este însã cã funcøia cuadraticã minimizează lungimea pãtratã a conexiunilor în locul lungimii liniare, şi de aceea nu corespunde direct cu scopul plasării liniare.

Li *et al.* [109] au propus utilizarea unei funcøii obiectiv spectrale care este un compromis între funcøia cuadraticã şi cea liniarã, ceea ce permite folosirea avantajelor oferite de ambele funcøii.

4.5 Metode neconvenționale de plasare

4.5.1 Plasarea prin algoritmi paraleli

Implementarea paralelã a metodei de cãlire simulatã nu este simplã, din cauza naturii secvenøiale a acestei metode. Cãlirea simulatã poate fi descrisã ca o secvenøã de lanøuri Markov omogene [142]: fiecare pas de calcul al unui lanø începe doar atunci când pasul precedent s-a terminat. Aceasta este condiøia ca întregul proces sã conducã la o configuraøie fezabilã unicã. Se pot utiliza douã tipuri de paralelism:

- Un paralelism pentru evaluarea fiecãrei mutãri: calculul unui anumit pas al lanøului Markov depinde numai de configuraøia sistemului înaintea acestui pas şi este executat fãrã interacøiune cu ceilaløi paşi. Astfel, evaluãrile diferitelor mutãri pot fi executate în paralel, ca şi calculele variaøiei funcøiei de cost şi a criteriului de acceptare. Acest tip de paralelism este dependent de problemã.
- Un paralelism global la nivelul lanøului Markov, care poate fi combinat cu primul tip de paralelism, dacã este necesar.

În literaturã au fost raportate diferite implementãri paralele ale metodei de cãlire simulatã [104] [141]. Diferenøele constau în principal în urmãtoarele aspecte:

- condiøiile de convergenøã ale algoritmului paralel;
- dependenøa paralelismului de problema care trebuie rezolvatã.

Pentru implementarea paralelã a problemei de plasare au fost sugerate diferite soluøii. Metoda utilizatã de Casotto *et al.* [37] constã în partiøionarea setului de celule care trebuie plasate într-un numãr de subseturi egal cu numãrul procesoarelor disponibile; fiecare subset fiind asignat unui anumit procesor. Procesoarele funcøionează asincron cât timp mutãrile apar într-un set dat de celule.

O altă abordare a problemei de plasare a celulelor a fost sugeratã de Darema, Kirkpatrick şi Norton. În acest caz, fiecare procesor evaluează o perturbaţie a lanţului Markov, cu condiţia cã douã procesoare nu pot muta aceleaşi celule simultan; astfel, nu existã conflict între procesoare şi configuraţia finalã este întotdeauna validã [142]. Atunci când o perturbaţie este acceptatã, configuraţia celulelor este actualizatã, indiferent de mutãrile care se calculează.

Roussel-Ragot şi Dreyfus [142] au sugerat o implementare paralelã a algoritmului de cãlire simulatã care, pe de o parte, este independentã de problemã, iar pe de altã parte, are aceleaşi proprietãţi de convergenţã ca şi algoritmul serial. Aceştia au utilizat douã moduri de paralelizare, în funcţie de valoarea temperaturii, şi au elaborat modele statistice care pot estima creşterea de vitezã pentru orice problemã, în funcţie de rata de acceptare şi numãrul de procesoare.

4.5.2 Plasarea prin reţele neuronale artificiale

Interesul recent manifestat pentru reţelele neuronale are la bazã recunoaşterea faptului cã creierul uman executã calcule într-un mod diferit de cel al calculatoarelor digitale. Calculatoarele pot fi extrem de rapide şi precise în executarea secvenţelor de instrucţiuni care au fost formulate în mod explicit. Un sistem uman de procesare a informaţiilor este format din neuroni cu o vitezã de comutare de aproximativ un milion de ori mai redusã decât cea a porţilor logice. Cu toate acestea, un sistem uman este mult mai eficient decât calculatoarele în rezolvarea unor probleme complexe din punct de vedere computaţional, ca de exemplu înţelegerea vorbirii.

4.5.2.1 Concepte de bază ale reţelelor neuronale artificiale

O reţea neuronalã artificialã poate fi definitã ca o reţea sinteticã care emulează reţelele neuronale biologice ale organismelor vii. O altã definiţie este cã reţelele neuronale artificiale reprezintã o clasã de algoritmi matematici, deoarece o reţea poate fi consideratã ca o notaţie graficã pentru o clasã largã de algoritmi [190].

O reţea neuronalã artificialã este un ansamblu al unui mare numãr de *neuroni artificiali*. Prima definiţie formalã a unui model de neuron artificial bazatã pe o considerare foarte simplificatã a unui neuron biologic a fost formulatã de McCulloch şi Pitts. Modelul McCullogh-Pitts al unui neuron este caracterizat prin formalism şi o definiţie matematicã precisã. Modelul utilizează însã unele simplificări importante: permite numai stări binare, operează cu presupunerea unui timp discret, şi presupune sincronismul funcţionării tuturor neuronilor dintr-o reţea de dimensiuni mari. Ponderile intrãrilor şi valorile de prag ale neuronilor sunt fixe.

Un neuron artificial constã dintr-un *element de procesare* (*nod*), care recepţionează un numãr de intrări analogice având conexiuni sinaptice, şi generează o singurã ieşire. Semnalele de intrare x_i ale neuronului sunt considerate unidirecţionale, ca şi semnalul de ieşire *out*. Fiecãrei intrări x_i i se asociază o pondere w_i.

Semnalul de ieşire al neuronului este dat de relaţia urmãtoare:

$$out = f(\mathbf{w}^{\mathrm{T}}\mathbf{x}), \text{ sau} \tag{4.72}$$

$$out = f\left(\sum_{i=1}^{n} w_i x_i\right) \tag{4.73}$$

unde **w** este vectorul ponderilor definit ca:

$$\mathbf{w} = [w_1 \;\; w_2 \;\; \ldots \;\; w_n]^{\mathrm{T}}$$

iar **x** este vectorul de intrare:

$$\mathbf{x} = [x_1 \;\; x_2 \;\; \ldots \;\; x_n]^{\mathrm{T}}$$

Ambii vectori sunt vectori coloanã. Funcøia $f(\mathbf{w}^T\mathbf{x})$ este numitã *funcøie de activare* a neuronului. Domeniul acestei funcøii este setul valorilor de activare, *net*, ale modelului neuronului, de aceea aceastã funcøie este utilizatã adesea ca $f(net)$. Variabila *net* este definitã ca un produs scalar al vectorului ponderilor øi al vectorului de intrare:

$$net = \mathbf{w}^T\mathbf{x} \tag{4.74}$$

Diferitele clase de reøele neuronale artificiale utilizeazã diferite funcøii $f(net)$. De asemenea, chiar în cazul aceleiaøi clase, neuronii se pot comporta diferit în timpul diferitelor faze ale funcøionãrii reøelei. De aceea, modelul general al neuronului este înlocuit de obicei cu un model specific, pentru o anumitã funcøie $f(net)$.

4.5.2.2 Rețele neuronale Hopfield

Reøelele neuronale Hopfield se pot utiliza pentru rezolvarea diferitelor probleme de optimizare, printre care øi problema de plasare, motiv pentru care sunt descrise pe scurt în aceastã secøiune. Pentru asemenea probleme, reøelele neuronale de tip gradient sunt cele mai adecvate. La aceste reøele, funcøia de energie descreøte în timp, timpul fiind presupus o variabilã continuã.

O reøea neuronalã Hopfield este caracterizatã ca o reøea puternic interconectatã de procesoare simple analogice. O asemenea reøea de tip gradient converge spre unul din minimele stabile din spaøiul stãrilor [114] [190]. Evoluøia sistemului este în direcøia generalã a gradientului negativ a unei funcøii de energie. În mod tipic, funcøia de energie a reøelei este echivalatã cu o anumitã funcøie obiectiv (penalizare) care trebuie minimizatã. Cãutarea unei energii minime executatã de o reøea de tip gradient corespunde cãutãrii unei soluøii a unei probleme de optimizare.

Avantajul reøelelor neuronale Hopfield este convergenøa rapidã la un minim stabil. Dacã la funcøia de energie se adaugã condiøii de restricøie cu o pondere mare, punctul iniøial va converge spre cel mai apropiat punct valid fãrã influenøã din partea funcøiei de cost.

4.5.2.3 Utilizarea rețelelor neuronale pentru plasare

Considerãm cazul cel mai simplu al problemei de plasare. Fiind date *n* module de circuit øi o matrice de conectivitate $\mathbf{C} = [C_{ij}]$, unde C_{ij} indicã conectivitatea între modulul *i* øi modulul *j*, cele *n* module trebuie plasate în *n* locaøii ale unei suprafeøe bidimensionale, astfel încât lungimea totalã de interconectare Manhattan sã fie minimizatã. Soluøia acestei probleme aparøine lui Yu [187], care a utilizat reøelele neuronale Hopfield pentru rezolvarea problemei de plasare. Acesta a modificat soluøia propusã de Hopfield øi Tank pentru rezolvarea problemei comis-voiajorului.

Se utilizeazã o reøea cu n^2 neuroni. Neuronii sunt numerotaøi de la 0 la $n^2 - 1$, de la stânga la dreapta øi de sus în jos. Valoarea unui element din poziøia (i, j) a matricii reprezintã "øansa" unui modul *i* de a fi poziøionat în locaøia *j*. Fiecare linie corespunde unui modul de circuit. Fiecare coloanã corespunde celor *n* locaøii posibile în care se pot plasa modulele. Pentru a obøine o soluøie fezabilã, un singur neuron din orice linie sau coloanã poate avea ieøirea 1. Ieøirile neuronilor sunt normalizate, astfel încât acestea au valori între 0 øi 1.

Urmãtoarea etapã este determinarea intensitãøii sinapselor. Se calculeazã mai întâi distanøa Manhattan între fiecare pereche de locaøii. Valoarea $T_{k_{i_1,j_1},l_{i_2,j_2}}$ a intensitãøii sinapsei între neuronii *k* øi *l* (care este un element al matricii intensitãøii sinapselor) este definitã ca øi conectivitatea între modulele de circuit i_1 øi i_2 multiplicatã cu $f(j_1, j_2)$, unde *f* este o funcøie a distanøei între locaøiile j_1 øi j_2, $k = i_1 \times \sqrt{n} + j_1$, iar $l = i_2 \times \sqrt{n} + j_2$. Dupã experimentãri funcøia *f* s-a ales ca fiind egalã cu (*offset* - distanøa Manhattan dintre j_1 øi j_2), unde parametrul *offset* este de obicei mai mare decât $\sqrt{n}$.

4.6 Algoritmi de plasare propuşi pentru circuitele FPGA cu resurse limitate de rutare

4.6.1 Algoritm de plasare pe baza tăieturii minime

4.6.1.1 Descrierea algoritmului de plasare

S-a propus un algoritm de plasare bazat pe metoda de partiţionare descrisă în capitolul 3, cu o funcţie obiectiv care urmăreşte pe lângă reducerea lungimii interconexiunilor, şi distribuirea uniformă a conexiunilor în cadrul partiţiilor. Ca urmare, rezultatul plasării maximizează posibilitatea unei rutări fezabile a circuitului. S-a studiat de asemenea efectul utilizării diferitelor secvenţe de aplicare a liniilor de tăieturã, cu scopul reducerii numărului de conexiuni în apropierea centrului circuitului, deoarece în cazul plasării obişnuite care se bazează pe metoda tăieturii minime apare în mod frecvent creşterea numărului de conexiuni în această zonă.

Dezavantajul algoritmilor de plasare care utilizează partiţionarea ierarhică este că dimensiunea tăieturii este singura metrică utilizată în cadrul funcţiei de cost. De aceea, este posibilă obţinerea unei dimensiuni reduse a tăieturii, şi în acelaşi timp a unor porţiuni cu un număr de conexiuni semnificativ diferit. Ca urmare, o asemenea plasare poate fi rutată într-un mod dificil. Pentru a elimina acest dezavantaj, este necesar ca în cadrul funcţiei de cost să se ia în considerare nu numai dimensiunea tăieturii, ci şi distribuţia interconexiunilor din cele două porţiuni ale bipartiţiei.

Algoritmul de plasare propus utilizează bipartiţionarea a cărei funcţie obiectiv urmăreşte minimizarea lungimii interconexiunilor simultan cu minimizarea diferenţei între numărul de conexiuni din cele două porţiuni. Această diferenţă este măsura care urmăreşte distribuţia echilibratã a conexiunilor din cele două porţiuni. În cadrul algoritmului de plasare se aplică în mod recursiv bipartiţionarea cu funcţia de cost descrisă în secţiunea 3.7.1, până când se ajunge la porţiuni care conţin o singură celulă a circuitului. Liniile de tăieturã se aplică în mod alternativ, pe orizontală şi pe verticală.

4.6.1.2 Secvenţa de aplicare a liniilor de tăietură

Pe lângă o procedură eficientă de partiţionare, este necesară o strategie adecvată de aplicare a liniilor de tăieturã. O secvenţă tradiţională de aplicare a liniilor de tăieturã, cum este procedura de partiţionare cuadraticã [146], are dezavantajul cã nu ţine cont de poziţia terminalelor externe. Pentru aceasta se poate utiliza tehnica numitã propagarea terminalelor. Această tehnică are însă dezavantaje. De exemplu, cele două regiuni sunt procesate secvenţial, neexistând criterii pentru stabilirea regiunii care trebuie procesată prima.

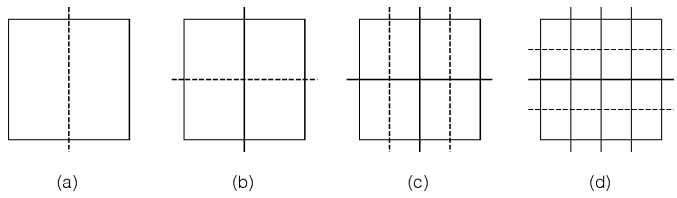

Figura 4.2. Alegerea liniei de tăieturã poziţionatã în centru.

Secvenţa de aplicare a liniilor de tăieturã are un rol important. În cadrul plasării convenţionale bazată pe tăietura minimă, se alege linia de tăieturã poziţionată în centrul regiunii curente. Dacă liniile de tăieturã se aplică în această ordine, Figura 4.2

indicã secvenøa liniilor de tãieturã, unde liniile întrerupte sunt cele aplicate recent, iar liniile continue sunt cele aplicate anterior.

Presupunem cã linia curentã de tãieturã, este în imediata apropiere a liniei de tãieturã din centru. Aceastã linie va înjumãtãøi patru regiuni øi conexiunile corespunzãtoare. În fiecare etapã a acestei bipartiøionãri, se pot interschimba perechi de noduri. Pentru a se øine cont de rezultatul bipartiøionãrii pentru liniile de tãieturã aplicate anterior, nu se permite interschimbarea nodurilor aflate în regiuni delimitate de liniile de tãieturã aplicate anterior. Numãrul de noduri dintr-o regiune este proporøional cu suprafaøa regiunii respective. Deoarece linia de tãieturã consideratã este în apropierea liniei de tãieturã din centru, regiunile intersectate au o suprafaøã redusã. În consecinøã, numãrul perechilor posibile care pot fi interschimbate în procesul de bipartiøionare este limitat. Aceasta are ca rezultat o dimensiune a tãieturii relativ ridicatã pentru liniile de tãieturã din apropierea centrului, din cauza numãrului mic de mutãri posibile.

Din cele de sus rezultã cã, pentru a se reduce dimensiune tãieturii în apropierea centrului, în aceastã zonã liniile de tãieturã trebuie aplicate în primele etape ale procesului de bipartiøionare, dupã cum se indicã în Figura 4.3.

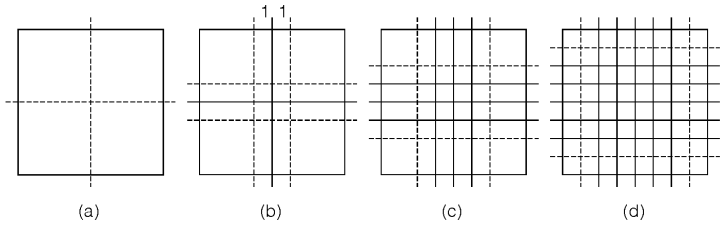

Figura 4.3. Secvenøa propusã pentru aplicarea liniilor de tãieturã.

4.6.2 Algoritm genetic pentru plasarea circuitelor FPGA

În timp ce aløi algoritmi îmbunãtãøesc în mod iterativ o plasare iniøialã prin mutarea unei singure celule sau prin interschimbarea a douã celule, un algoritm genetic pornešte de la un set de plasãri iniøiale care reprezintã populaøia iniøialã. Algoritmul încearcã sã combine caracteristicile adecvate din douã configuraøii de plasare pentru a forma o nouã plasare. Aceste caracteristici adecvate se numesc scheme, ele fiind plasãrile relative ale subseturilor de celule. Prin procesul de evoluøie simulatã, dupã un numãr de generaøii, soluøiile candidate reøin caracteristicile mai bune ale soluøiilor multiple din generaøiile anterioare. Acest paralelism intrinsec conferã un avantaj algoritmilor genetici faøã de metoda cãlirii simulate [102], care utilizeazã o singurã soluøie.

Datoritã acestor avantaje, s-a încercat rezolvarea problemei de plasare a circuitelor FPGA printr-un algoritm genetic. Dupã cunoštinøele noastre, nu a fost publicat pânã în prezent un algoritm genetic pentru plasarea circuitelor FPGA. Algoritmul descris în aceastã secøiune, implementat pentru seria de circuite *Atmel* 6000, reprezintã deci o primã contribuøie în acest sens.

O plasare este acceptabilã dacã se poate realiza rutarea în proporøie de 100%. Aceasta nu este o sarcinã simplã pentru arhitecturile FPGA cu resurse limitate de rutare. O plasare corespunzãtoare va realiza nu numai gruparea blocurilor conectate, dar va asigura ca blocurile logice sã nu fie plasate foarte apropiat, pentru a se permite rutarea circuitului. O altã contribuøie o reprezintã un algoritm care alocã un numãr de celule libere în cadrul procesului de plasare, în scopul utilizãrii acestor celule pentru rutare. Funcøia de cost utilizatã optimizeazã un numãr de diferite metrici, care cuprind atât lungimea interconexiunilor, cât øi mãsuri ale rutabilitãøii plasãrii.

4.6.2.1 Utilizarea algoritmilor genetici pentru plasare

Deşi nu au fost utilizaţi pentru plasarea circuitelor FPGA, algoritmii genetici au fost utilizaţi pentru plasarea circuitelor VLSI. Lucrarea clasicã a fost realizatã de Cohoon *et al.* [57], [58]. Aceştia au codificat o plasare prin notaţia polonezã a unui arbore binar, un cromozom fiind deci reprezentat printr-un şir. Au fost utilizaţi diferiţi operatori de recombinare, care opereazã fie direct asupra şirurilor, fie iau în considerare structura de arbore prin decodificarea cromozomului.

Esbensen [73] a descris un algoritm genetic pentru plasarea macro-celulelor în care reprezentarea genotipului este de asemenea un arbore binar. Spre deosebire de abordarea lui Cohoon *et al.*, acest arbore nu caracterizeazã direct o plasare, ci aceasta poate fi generatã prin decodificarea arborelui. Operatorii genetici lucreazã direct cu structura arborelui. Esbensen şi Mazumder [74] au raportat un algoritm numit SAGA, care este o generalizare a algoritmului genetic şi a algoritmului de cãlire simulatã. În funcţie de setarea parametrilor sãi de control, SAGA se comportã ca un algoritm genetic, un algoritm de cãlire simulatã, sau o combinaţie a acestora. Autorii au arãtat experimental cã prin mixarea algoritmului genetic cu algoritmul de cãlire simulatã se obţine o plasare de calitate mai bunã decât printr-un algoritm pur genetic.

Mohan şi Mazumder [124] au descris un algoritm genetic distribuit pentru plasarea celulelor standard. Algoritmul a fost elaborat pentru a fi executat pe o reţea de staţii de lucru. Procedura de plasare distribuitã executã un algoritm genetic de bazã pe fiecare procesor din reţea. Este introdus un nou operator genetic, *migraţia*, care transferã informaţii de plasare de la un procesor la altul în cadrul reţelei.

Schnecke şi Vornberger [150] au prezentat un algoritm genetic pentru proiectarea fizicã a circuitelor VLSI. Algoritmul combinã etapele de amplasare şi de rutare, optimizând simultan plasarea celulelor şi rutarea. Aceiaşi autori au descris în [149] un algoritm genetic paralel pentru optimizarea combinatã a plasării şi a rutării. Ambii algoritmi [149], [150] utilizeazã un genotip codificat ca un arbore binar, care defineşte plasarea relativã a celulelor.

4.6.2.2 Reprezentarea soluției

Reprezentarea soluţiei are un rol major în proiectarea unui algoritm genetic eficient. Pentru soluţia problemei de plasare s-a ales reprezentarea printr-un şir de înregistrãri. Fiecare înregistrare reprezintã o celulã, conţinând identificatorul celulei şi coordonatele (*x*, *y*) ale acesteia. Poziţia înregistrării corespunzãtoare unei celule nu determinã întotdeauna poziţia fizicã a celulei în circuit. Totuşi, în anumite puncte din algoritm, poziţia celulei se recalculeazã pe baza ordonării înregistrãrilor celulelor în şir.

4.6.2.3 Operatori genetici

Încrucişarea este principalul operator genetic. Încrucişarea combinã scheme din ambii pãrinţi, şi astfel urmaşul moşteneşte unele din caracteristicile pãrinţilor. Cea mai simplã formã de încrucişare nu poate fi aplicatã în acest caz, deoarece poate crea şiruri care nu au un corespondent fizic. În algoritmul descris, s-a utilizat un operator numit încrucişare mapatã parţial (*PMX*).

Încrucişarea *PMX* se executã astfel (Figura 4.4): se selecteazã doi pãrinţi (1 şi 2) şi se alege un punct de tãieturã arbitrar. Ca şi în cazul încrucişării simple, întregul subşir din dreapta al pãrintelui 2 este copiat în şirul urmaşului. Apoi, subşirul din stânga al pãrintelui 1 este parcurs genã cu genã, de la stânga pânã la punctul de tãieturã. Dacã o genã nu existã în cadrul urmaşului, este copiatã în şirul acestuia. Dacã însã gena existã deja în cadrul urmaşului, este determinatã poziţia sa în pãrintele 2, şi este copiatã gena din poziţia determinatã a pãrintelui 1.

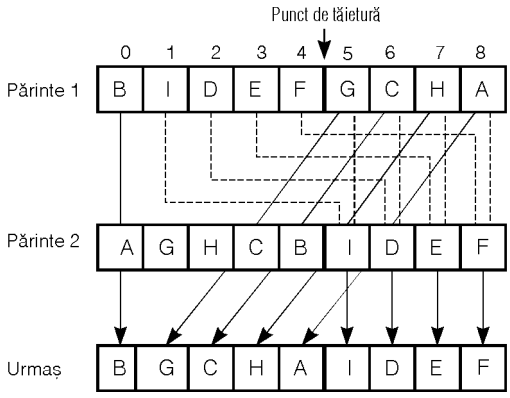

Figura 4.4. Încrucișarea *PMX*.

Mutaøia produce modificări aleatoare incrementale ale urmașului generat prin încrucișare. Pentru problema de plasare, ca tehnicã de mutaøie se poate utiliza interschimbarea perechilor. Operatorul de mutaøie generează noi tripleøi celulã-coordonate. În cazul în care aceștia se comportã în mod corespunzãtor, configuraøiile care le conøin vor fi reøinute. Mutaøia este controlatã de un parametru numit rata de mutaøie M_r. Procesul de mutaøie va permite ca diversitatea populaøiei sã fie manøinutã în etapele finale ale algoritmului genetic. Mutaøia permite de asemenea ca algoritmul genetic sã evite minimele locale.

Inversiunea este o operaøie care modificã lungimea efectivã a unei scheme fãrã a altera viabilitatea individului în scopul creșterii probabilitãøii de supravieøuire a schemelor mai lungi. Operatorul de inversiune modificã poziøiile înregistrãrilor celulelor în șirul care reprezintã o plasare, dar nu modificã poziøia fizicã a celulelor din circuit. Inversiunea este controlatã de un parametru numit rata de inversiune I_r.

4.6.2.4 Selecția populației pentru generația următoare

Algoritmii genetici convenøionali utilizează metoda prin care se generează doi urmași din doi pãrinøi și se înlocuiesc pãrinøii prin cei doi urmași pentru prelucrãrile din generaøia urmãtoare. În algoritmul propus s-a utilizat însã o strategie diferitã. Dupã generarea urmașilor, aceștia înlocuiesc un numãr echivalent de indivizi din populaøia curentã, aleși dintre cei cu viabilitatea cea mai redusã. Această strategie permite supravieøuirea soluøiilor mai bune de-a lungul unui mare numãr de generaøii. Soluøiile generate prin operatorii genetici concurează pentru cel puøin o generaøie. Această metodã de selecøie asigurã performanøe corespunzãtoare ale algoritmului genetic.

4.6.2.5 Aspecte specifice pentru circuitele FPGA cu resurse limitate de rutare

Pentru circuitele FPGA cu resurse limitate de rutare, încã în etapa de plasare sunt necesare mãsuri speciale pentru a se asigura rutabilitatea circuitelor. Pentru a se asigura rutabilitatea plasãrii, ideea este de a se utiliza un numãr de celule libere pentru rutare. Aceste celule sunt alocate în cadrul procesului de plasare.

În timp ce minimizarea lungimii totale a interconexiunilor reduce numãrul resurselor de rutare necesare în mod global, nu poate asigura faptul cã semnalele vor putea fi rutate local, datoritã resurselor limitate de rutare. Pentru rezolvarea acestei probleme, au fost adãugate douã componente la funcøia de cost. Componenta de rutabilitate localã atribuie o penalizare acelor situaøii în care se poate determina faptul cã

o conexiune nu poate fi rutată utilizând resurse locale de rutare. Rutabilitatea locală poate depista numai unele plasări ilegale care pot fi determinate din contextul imediat.

Componenta de echilibrare a densităţii este prevăzută pentru a se preveni congestia rutării datoritã unei concentrări ridicate a funcţiilor într-o zonă a circuitului. Aceasta se realizează prin considerarea unor ferestre de celule şi contorizarea intrărilor utilizate în această regiune. Pentru a se asigura distribuţia uniformă a celulelor libere, penalizarea utilizată este pătratul numărului de intrări utilizate peste un anumit prag într-o fereastră, valori care se însumează pentru toate ferestrele. Se examinează fiecare fereastră din circuit, astfel încât ferestrele se suprapun. Este permisă deplasarea ferestrelor dincolo de limitele circuitului, pentru celulele virtuale aflate dincolo de marginile circuitului presupunându-se un număr de intrări utilizate egal cu media globală.

4.6.3 Rezultate experimentale

Algoritmul de plasare propus care utilizează secvenţa propusă a liniilor de tăieturã a fost implementat în limbajul C. Experimentele au fost efectuate pe un calculator IBM PC cu un procesor Pentium de 133 MHz, sub sistemul de operare Windows NT Version 4.0. S-a utilizat un număr de nouă circuite de test din cadrul setului de circuite al centrului *MCNC* (*Microelectronics Center of North Carolina*).

S-a comparat suma dimensiunilor tăieturilor pentru toate liniile de tăieturã aplicate pentru algoritmul propus şi algoritmul tradiţional. Comparaţia s-a efectuat pentru două cazuri, atunci când nu se urmăreşte echilibrarea numărului de conexiuni din cele două partiţii, şi atunci când se realizează şi echilibrarea numărului de conexiuni. Rezultatele aratã cã prin algoritmul propus s-a obţinut o reducere a sumei tăieturilor cu un procent cuprins între 20 % şi 43.3 % pentru cazul în care nu se urmăreşte echilibrarea numărului de conexiuni. În acest caz, reducerea medie este de 32 %. Pentru cazul în care se urmăreşte şi echilibrarea numărului de conexiuni, reducerea obţinutã este cuprinsă între 8.6 % şi 39.2 %, reducerea medie fiind de 20.6 %.

De asemenea, s-a comparat valoarea maximă a tăieturii pentru toate liniile de tăieturã aplicate pentru algoritmul propus şi algoritmul tradiţional. Comparaţia s-a efectuat pentru aceleaşi două cazuri. Prin algoritmul propus s-a obţinut o reducere a tăieturii maxime cu un procent de până la 30 % pentru cazul în care nu se urmăreşte echilibrarea numărului de conexiuni. În acest caz, reducerea medie este de 17.75 %. Pentru cazul în care se urmăreşte şi echilibrarea numărului de conexiuni, reducerea obţinutã este de până la 35.7 %, reducerea medie fiind de 22.85 %. Deci, prin algoritmul propus se obţine atât o reducere a sumei tăieturilor, cât şi o reducere a valorii maxime a tăieturii.

Algoritmul genetic pentru plasare a fost implementat în limbajul C. Experimentele au fost efectuate utilizând de asemenea circuite de test din setul *MCNC*. S-a studiat efectul dimensiunii populaţiei şi a ratei de mutaţie asupra calităţii rezultatelor în scopul determinării unor valori optime pentru acești parametri. Calitatea rezultatelor a fost apreciată pe baza lungimii totale a interconexiunilor. Pe baza experimentelor, au fost utilizate următoarele valori ale parametrilor algoritmului genetic: dimensiunea populaţiei N_p = 80, rata de mutaţie M_r = 0.05, rata de inversiune I_r = 0.15, numărul de generaţii N_g = 1000.

4.7 Concluzii

În acest capitol a fost studiată problema de plasare a modulelor, având ca principal obiectiv asigurarea rutabilităţii circuitelor. Au fost prezentate principalele funcţii de cost şi restricţii utilizate pentru rezolvarea acestei probleme. Funcţia obiectiv cea mai des utilizată în cadrul plasării este lungimea totală estimată a conexiunilor.

Metodele de plasare bazate pe partiøionare implicã aplicarea recursivã a unui algoritm de partiøionare. Deši metoda de cãlire simulatã poate obøine soluøia optimã globalã, timpul de calcul necesar pentru circuite de dimensiuni mari este foarte mare. Pentru a elimina acest dezavantaj, au fost propuse mai multe tehnici de creštere a vitezei pentru acest algoritm.

O altã metodã care a fost descrisã utilizeazã partiøionarea ierarhicã pe baza tãieturii proporøionale. O posibilitate de rezolvare a problemei de plasare este transformarea acesteia într-o problemã de optimizare numericã. A fost descrisã plasarea controlatã de forøe, în care problema de plasare este redusã la soluøionarea unui sistem de ecuaøii liniare pentru a determina locaøiile de echilibru ale celulelor. Metodele spectrale au fost utilizate ši pentru problema de plasare. A fost descrisã o metodã de plasare liniarã propusã de Li care utilizeazã o funcøie obiectiv spectralã constând dintr-un compromis între funcøia cuadraticã ši cea liniarã, ceea ce permite folosirea avantajelor oferite de ambele funcøii. Dintre metodele neconvenøionale de plasare, a fost prezentatã plasarea prin algoritmi paraleli ši plasarea prin reøele neuronale artificiale.

În acest capitol s-au propus doi algoritmi de plasare pentru circuitele FPGA cu resurse limitate de rutare. Primul algoritm se bazeazã pe algoritmul de partiøionare propus în capitolul 3. Algoritmul de plasare propus utilizeazã bipartiøionarea a cãrei funcøie obiectiv urmãreste minimizarea lungimii interconexiunilor simultan cu distribuøia echilibratã a conexiunilor din cele douã porøiuni. A fost propusã de asemenea o secvenøã de aplicare a liniilor de tãieturã care este mai eficientã decât secvenøele tradiøionale. Rezultatele experimentale aratã cã prin aplicarea acestei secvenøe de tãieturã se obøine atât o reducere a dimensiunii maxime a tãieturii, cât ši o reducere a sumei totale a tãieturilor, comparativ cu procedura de plasare cuadraticã.

Al doilea algoritm propus este un algoritm genetic pentru plasarea circuitelor FPGA, dupã cunoštinøele noastre nefiind publicat pânã în prezent un algoritm genetic pentru plasarea acestor circuite. Pentru facilitarea rutãrii, algoritmul alocã un numãr de celule libere în cadrul procesului de plasare, în scopul utilizãrii acestor celule pentru rutare. Algoritmul utilizeazã o funcøie de cost care optimizeazã diferite metrici, care cuprind atât lungimea interconexiunilor, cât ši mãsuri ale rutabilitãøii plasãrii. Funcøia de cost cuprinde o componentã de estimare a rutabilitãøii locale ši o componentã de echilibrare a densitãøii.

5. RUTAREA CIRCUITELOR CU RESURSE LIMITATE DE RUTARE

5.1 Introducere

În procesul de proiectare automatã a circuitelor VLSI sau de implementare a sistemelor digitale prin circuite FPGA, etapa urmãtoare celei de plasare a modulelor este *rutarea*. Rutarea necesitã în jur de 30% din timpul de proiectare, iar interconexiunile necesitã un procent ridicat din suprafaøa circuitelor [146]. Primii algoritmi de rutare automatã au fost dezvoltaøi pentru proiectarea circuitelor imprimate. Unele idei de bazã ale rutãrii automate rezultate de aici sunt valide în continuare, fiind adaptate pentru circuitele VLSI ši circuitele FPGA.

Pentru rutare se adoptã de obicei o abordare în douã etape: se executã mai întâi *rutarea globalã*, urmatã apoi de *rutarea detaliatã*. Obiectivul rutãrii globale este de a se elabora un plan de rutare astfel încât fiecare conexiune sã fie asignatã unor regiuni particulare de rutare, în timp ce se încearcã minimizarea unei funcøii obiectiv date. Rutarea detaliatã se aplicã apoi pentru fiecare regiune de rutare, ši fiecãrei conexiuni i se asigneazã piste particulare de rutare.

5.2 Definirea problemei de rutare

Fiind dat un set de celule şi porturile acestora (pinii de intrare, ieşire, ceas, alimentare şi masă), un set de conexiuni (seturi de puncte care trebuie conectate împreunã din punct de vedere electric), şi locaøiile celulelor (obøinute în urma procesului de plasare), rutarea constă în determinarea căilor adecvate pentru interconexiunile dintre seturile de pini. Aceste căi adecvate minimizează funcøia obiectiv dată, supusă unor restricøii. Restricøiile pot fi impuse de proiectant, de procesul de implementare, de tipul circuitului sau de stilul de proiectare.

5.3 Funcții de cost şi restricții

5.3.1 Funcții de cost şi restricții pentru rutarea globală

Rutarea globală este diferită pentru diferitele tipuri de circuite. În cazul *reøelelor de porøi*, regiunile de rutare constau din canale orizontale şi verticale. Canalele sunt regiuni dreptunghiulare cu pini amplasaøi pe marginile opuse ale regiunii. Capacitãtile de rutare disponibile în cadrul canalelor sunt fixe. O soluøie de rutare globală nu trebuie să depăşească capacitãøile canalelor.

Pentru circuitele cu *celule standard*, regiunile de rutare sunt canale orizontale cu pini amplasaøi pe marginile de sus şi de jos ale canalelor. Rutarea globală constă în asignarea conexiunilor acestor canale astfel încât să se minimizeze congestia canalelor şi lungimea totală a conexiunilor. Rutarea între canale este asiguratã prin celule de trecere între rândurile de celule. Canalele nu au capacitãøi fixate în prealabil.

În cazul circuitelor cu *macro-celule*, dimensiunea şi forma celulelor este variată. Aceasta conduce la regiuni de rutare neregulate. Aceste regiuni pot fi descompuse în canale orizontale şi verticale, şi uneori blocuri de comutare (regiuni dreptunghiulare cu pini pe toate cele patru margini). Identificarea acestor regiuni este un prim pas esenøial pentru rutarea globală. Nici în acest caz, regiunile de rutare nu au capacitãøi fixate.

Atât pentru circuitele cu celule standard, cât şi pentru cele cu macro-celule, obiectivul rutării globale este minimizarea spaøiului de rutare necesar şi a lungimii totale a conexiunilor, asigurând în acelaşi timp succesul rutării detaliate ulterioare. Funcøia de cost este de aceea o măsură a spaøiului total de rutare. Restricøiile pot fi o limită a numărului maxim de piste pe canal şi/sau restricøii asupra performanøelor.

5.3.2 Funcții de cost şi restricții pentru rutarea detaliată

Obiectivul principal al rutării detaliate este obøinerea rutării complet automate, fără intervenøie manuală sau cu o intervenøie minimă. Pentru implementarea unui circuit într-un spaøiu minim, este esenøială reducerea spaøiului ocupat de interconexiuni. Lungimea conexiunilor individuale trebuie de asemenea redusă pentru a se satisface criteriile de performanøă. Deci, obiectivul celor mai multor programe de rutare este realizarea rutării complete utilizând conexiuni de lungime cât mai redusă.

Performanøele sunt afectate datoritã întârzierilor de semnal. Întârzierile introduse de porøile logice s-au redus considerabil, astfel încât întârzierile datorate interconexiunilor nu mai pot fi ignorate. Obiectivul programului de rutare este deci nu numai reducerea lungimii totale a conexiunilor, dar şi păstrarea întârzierii maxime a fiecărei conexiuni sub o anumită valoare limită.

Algoritmii de rutare detaliată trebuie să øină cont de un set dat de restricøii. Principalele tipuri de restricøii sunt: a) restricøii de plasare, b) numărul straturilor de rutare, şi c) restricøii geometrice.

5.4 Sinteza metodelor de rutare

5.4.1 Prezentarea sintetică a metodelor de rutare globală

În cazul *rutãrii globale*, metodele raportate în literaturã se pot împãrøi în patru categorii principale: 1) metode secvenøiale; 2) metode de cãutare aleatoare; 3) metoda programãrii liniare; 4) metoda descompunerii ierarhice.

O etapã importantã de pregãtire a rutãrii globale este *determinarea regiunilor de rutare*. Cai øi Otten [34] au descris un algoritm pentru conversia joncøiunilor de tip + în joncøiuni de tip T în cazul circuitelor cu macro-celule. Cai øi Wong [35] au elaborat un algoritm pentru definirea regiunilor de rutare ca øi canale dreptunghiulare øi blocuri de comutare.

O altã problemã importantã întâlnitã în timpul rutãrii globale, ca øi la rutarea detaliatã, este cea a construirii unui *arbore Steiner* pentru fiecare conexiune. Existã numeroase lucrãri asupra acestui subiect, fiind descrise metode euristice pentru gãsirea unui arbore Steiner apropiat de cel optim, într-un timp rezonabil [60] [99] [148].

Chiang *et al.* [54] au propus utilizarea *arborilor Steiner ponderaøi* pentru rutarea globalã. Aceeaøi autori au descris în [53] o metodã de rutare globalã bazatã pe *arbori Steiner min-max*, care au muchiile de pondere maximã minimizate. Conexiunile sunt ordonate mai întâi pe baza criteriilor de lungime, multiplicitate øi importanøã. Pentru fiecare conexiune se construieøte apoi un arbore Steiner min-max.

Metoda de cãlire simulatã a fost utilizatã øi pentru rutarea globalã, mai întâi de Vecchi øi Kirkpatrick, iar apoi de Sechen øi Sangiovanni-Vincentelli [151], în cadrul pachetului de programe TimberWolf. Acest pachet este destinat pentru plasarea øi rutarea globalã a circuitelor cu celule standard.

Pentru rutarea circuitelor VLSI au fost utilizate metode de *programare liniarã*. Vannelli [172] a descris o modificare a metodei punctului interior a lui Karmakar pentru rezolvarea problemei de rutare globalã. Spre deosebire de abordãrile bazate pe metoda simplex, metoda propusã permite reducerea semnificativã a dimensiunii problemei pe mãsura evoluøiei algoritmului.

În scopul reducerii complexitãøii problemei de rutare globalã, au fost propuse *metode ierarhice*, care efectueazã o descompunere ierarhicã a problemei în mai multe subprobleme, acestea fiind soluøionate individual. Soluøiile parøiale sunt combinate apoi pentru a produce o soluøie a problemei globale originale. Metodele propuse independent de Marek-Sadowska øi Lauther sunt similare, soluøiile generate fiind superioare celorlalte metode ierarhice raportate [146].

Viteza circuitelor este influenøatã în mare mãsurã de întârzierile datorate interconexiunilor. Aceasta a determinat apariøia unor sisteme de proiectare în care accentul se pune pe *maximizarea performanøelor*, în special pe reducerea întârzierilor de interconectare. Pentru a se evita întârzierile mari pe cãile critice, au existat încercãri pentru a controla etapa de rutare globalã de cerinøele de timp ale circuitelor. Jackson, Kuh øi Marek-Sadowska au raportat o metodã ierarhicã pentru rutarea controlatã de restricøiile de timp [60].

Cong *et al.* [60] au raportat o metodã de rutare globalã cu minimizarea simultanã a lungimii totale de interconectare øi a întârzierii maxime de interconectare. Aceastã metodã este bazatã arbori de rutare cu razã limitatã, pentru construcøia acestora utilizându-se euristici bazate pe algoritmul lui Prim pentru arbori de acoperire minimi. Aceeaøi autori au elaborat o metodã care minimizeazã simultan lungimea totalã de interconectare øi întârzierea maximã (indicatã de raza arborelui).

Shragowitz øi Keal au formulat rutarea globalã cu ajutorul unui model de *reøea de flux* [146]. Soluøia este obøinutã în douã etape. În prima etapã, este soluøionatã problema fãrã a øine cont de restricøii. În a doua etapã, se aplicã o procedurã iterativã

în care, la fiecare iteraøie, este rerutatã conexiunea pentru care rezultã o reducere maximã a fluxului.

În mod tradiøional, rutarea globalã a fost utilizatã pentru elaborarea unui plan de rutare care va fi executat de rutarea detaliatã. O altã utilizare a rutării globale este pentru *verificarea soluøiilor obøinute în urma plasării*. Rutarea globalã este utilizatã pentru controlul procedurii de plasare în sensul producerii unei plasări rutabile. Una din lucrările elaborate în acest sens se datorează lui Shragowitz *et al.*, în care se descrie un algoritm constructiv de plasare controlat de rutarea globalã [146].

Pentru rezolvarea problemei de rutare globalã au fost utilizate şi *metode netradiøionale:* reøele neuronale artificiale, algoritmi paraleli, evoluøia simulatã. Shih *et al.* [154] au utilizat o reøea neuronalã artificialã cu douã straturi. Un strat este utilizat pentru minimizarea lungimii conexiunilor şi pentru obøinerea unei distribuøii uniforme a conexiunilor în cadrul canalelor de rutare. Al doilea strat este utilizat pentru a impune restricøiile privind capacitãøile canalelor.

Rutarea este în general un proces mare consumator de timp. Majoritatea soluøiilor propuse permit o implementare paralelã. Rose [140] a descris un algoritm de rutarea globalã pentru celule standard, *LocusRoute*, şi implementarea paralelã a acestui algoritm. În implementarea paralelã raportatã, Rose a descris douã strategii pentru paralelizarea procesului de rutare. Prima strategie constã în rutarea mai multor conexiuni în paralel. În a doua strategie, se evaluează în paralel mai muløi arbori de rutare pentru fiecare conexiune.

5.4.2 Prezentarea sintetică a metodelor de rutare detaliată

Rutarea detaliatã poate fi împărøitã în *rutare detaliatã generalã* şi *rutare detaliatã cu restricøii*. Rutarea detaliatã cu restricøii pote fi împărøitã la rândul ei în rutare prin *canale* şi rutare prin *blocuri de conexiune*.

Una din metodele cele mai utilizate de *rutare detaliatã generalã* este *rutarea labirint*, algoritmul cel mai cunoscut fiind cel elaborat de Lee. Dezavantajul acestui algoritm este necesarul ridicat de memorie şi timp de execuøie. Pentru eliminarea acestor dezavantaje au fost propuse diferite tehnici. Hadlock a sugerat o nouă tehnicã de etichetare a celulelor bazatã pe numere de deturnare [146]. Soukup a sugerat adãugarea unei cãutări în adâncime.

O altã metodã de rutare detaliatã generalã este rutarea prin *cãutarea liniilor*. Prin aceastã metodã se eliminã necesarul ridicat de memorie al algoritmului Lee, deoarece spaøiul de rutare este reprezentat prin segmente de linii. Spre deosebire de algoritmul Lee, în acest caz se efectuează cãutarea în adâncime. Principalii algoritmi au fost propuşi de Mikami şi Tabuchi, respectiv Hadlock.

Pentru *rutarea prin canale*, Deutch a propus un algoritm pentru evitarea buclelor în graful constrângerilor verticale şi pentru reducerea densitãøii canalului. Aceasta se realizează prin divizarea segmentelor orizontale ale unei conexiuni, cu efectul minimizării numărului de piste orizontale. Divizarea orizontalã a unei conexiuni poate fi realizatã numai în poziøiile terminalelor. Algoritmul Deutch divizează fiecare conexiune multipin în segmente orizontale individuale.

O euristicã de tip *greedy* pentru rutarea prin canale a fost propusã de Rivest şi Fiduccia. Acest algoritm rutează canalul coloanã cu coloanã începând din stânga. În fiecare coloanã algoritmul aplicã o secvenøã de euristici inteligente pentru maximizarea numărului de piste disponibile în următoarea coloanã. Nu se utilizează constrângeri orizontale sau verticale, deciziile fiind luate local, la nivelul coloanei.

Un algoritm de rutare prin canale bazat pe *sortare* a fost propus de Chaudry şi Robinson [47]. Această metodã presupune cã interconexiunile pot fi rutate nu numai orizontal şi vertical, ci şi la 45° şi 135°. Algoritmul utilizează sortarea bubble.

O euristică pentru rutarea prin canale care asignează interconexiunile pistă cu pistă într-un mod *greedy* a fost propusă de Ho *et al.* [92]. Structura de date și strategia utilizată sunt simple și pot fi generalizate pentru obținerea unei clase de euristici pentru rutarea prin canale. Acest algoritm are posibilitatea de revenire prin care cresc șansele de efectuare a rutării complete cu un număr minim de piste.

5.5 Rutarea globală

5.5.1 Regiuni de rutare

Definirea regiunilor de rutare constă în partiționarea spațiului de rutare într-un set de regiuni dreptunghiulare care nu se intersectează, numite *canale*. Pot exista două tipuri de canale: orizontale și verticale. În cele mai multe cazuri, canalele orizontale și verticale pot veni în contact prin intersecții în formă de T. Definirea și ordonarea canalelor este o parte esențială a procesului de proiectare fizică, reprezentând legătura între plasare, rutarea globală și rutarea detaliată.

5.5.2 Rutarea globală secvențială

Rutarea globală secvențială este metoda cea mai des utilizată. După ce au fost identificate canalele de rutare și a fost construit graful de rutare corespunzător, rutarea globală se continuă astfel. Pentru fiecare conexiune, se marchează vârfurile grafului de conectivitate al canalelor în care conexiunea respectivă are pini. Deci, rutarea conexiunii se reduce la identificarea unui arbore care acoperă vârfurile marcate.

Dacă conexiunea are pini doar în două vârfuri ale grafului, problema se reduce la determinarea căii celei mai scurte între vârfurile marcate. Dacă graful este un graf de caroiaj, se poate utiliza algoritmul lui Lee. Pentru toate cele trei modele de grafuri, se poate utiliza algoritmul lui Dijkstra pentru determinarea căii celei mai scurte.

În general, conexiunile au mai mult de doi pini. Determinarea căii celei mai scurte care acoperă trei sau mai multe noduri constituie problema arborelui Steiner.

5.5.2.1 Problema arborelui Steiner

Fie *M* un set de vârfuri marcate. Un arbore care conectează toate vârfurile din *M* ca și alte vârfuri din *G* care nu sunt în *M* este numit un *arbore Steiner*. Un *arbore Steiner minim* este un arbore Steiner cu o lungime minimă. Problema arborelui Steiner este NP-completă. În literatură au fost publicate diferite metode euristice pentru determinarea arborelui Steiner. Cele mai multe dintre aceastea utilizează o variantă modificată a algoritmului Dijkstra pentru determinarea căii celei mai scurte sau o variație a algoritmului Lee pentru rutarea de tip labirint. De obicei, o asemenea euristică este de tip greedy și se execută astfel. Se selectează mai întâi unul din vârfurile marcate. Se identifică apoi calea cea mai scurtă la oricare din vârfurile marcate rămase. Apoi este selectat unul din vârfurile marcate rămase și este identificată calea cea mai scurtă de la acest vârf la oricare din vârfurile arborelui parțial. Acest proces continuă până când se procesează toate vârfurile marcate.

5.5.2.2 Rutarea globală prin metoda parcurgerii labirintului

Primul pas al rutării globale este modelarea regiunilor de rutare. Zonele orizontale și verticale de rutare sunt definite prin extinderea muchiilor orizontale și verticale ale celulelor plasate până la marginea suprafeței. Regiunile de rutare ale acestui model reprezintă intersecțiile dintre zonele orizontale și verticale de rutare. În acest caz, nu se garantează ca regiunile de rutare să fie canale.

După identificarea canalelor (regiunilor) de rutare, etapa următoare este asignarea de conexiuni acestor regiuni. Pentru aceasta, canalele sunt modelate printr-un *graf de conectivitate al canalelor*. Pentru rutarea cu două straturi, fiecărui nod i se asignează două ponderi reprezentând capacitatea orizontală (lățimea) și capacitatea verticală (lungimea) canalului corespunzător.

În cazul rutării globale *independente de ordine*, fiecare conexiune este rutată independent de toate celelalte conexiuni. Se identifică apoi zonele congestionate și conexiunile afectate sunt rerutate, penalizând căile care trec prin aceste zone. Această metodă nu necesită ordonarea conexiunilor și reduce în mod considerabil complexitatea spațiului de căutare, deoarece singurele obstacole sunt celulele. Totuși, poate fi necesar un număr mare de iterații pentru găsirea unei soluții fezabile.

În cazul rutării globale *dependente de ordine*, mai întâi se ordonează conexiunile conform anumitor criterii. Apoi conexiunile sunt rutate în ordinea rezultată, actualizând spațiul de rutare disponibil după fiecare conexiune. Căutarea este mai complexă, deoarece numărul de obstacole este mai mare. Ambele metode sunt oarecum similare prin faptul că ele încearcă identificarea unui arbore Steiner pentru o conexiune la un moment dat.

5.5.2.3 Rutarea globală utilizând arbori Steiner ponderați

Fie un set de conexiuni cu terminale multiple $\eta = \{N_1, \ldots, N_n\}$ pentru care trebuie executată rutarea globală. Presupunem că suprafața de rutare este divizată în regiuni dreptunghiulare. Fiecare conexiune N cu k terminale este specificată printr-un k-tuplu $(R_1, \ldots, R_k)$, unde R_i, $1 \le i \le k$, sunt regiunile conținând terminalele conexiunii N. Regiunile R_i nu sunt neapărat distincte, deoarece o conexiune poate avea mai multe terminale într-o regiune. Pentru rutarea globală, pentru fiecare conexiune trebuie specificată o secvență de regiuni prin care va trece conexiunea respectivă.

Se consideră o divizare a planului într-o colecție de regiuni $\mathbb{R} = \{R_1, \ldots, R_m\}$. Regiunii R_i îi este asignată o pondere pozitivă w_i. Se consideră o cale $\mathbb{P}$ care conectează două puncte p_a și p_b, trecând prin diferite regiuni. Se notează cu ℓ_i lungimea căii $\mathbb{P}$ în regiunea R_i, deci $\ell_i = |\mathbb{P} \cap R_i|$. Ponderea căii $\mathbb{P}$ este $W(\mathbb{P}) = \Sigma\, \ell_i\, w_i$.

Fiind dat un set de puncte P din $\mathbb{R}$, problema este de a obține un arbore Steiner cu pondere minimă care interconectează punctele din P. Se consideră numai căi rectiliniare. Aceasta este problema *arborelui Steiner rectilinar ponderat*. Pentru a găsi o cale cu pondere minimă între două puncte p_a și p_b din $\mathbb{R}$, se construiește un graf G, numit *graful de căutare* al $\mathbb{R}$. Acest graf se obține prin extinderea marginilor fiecărei regiuni până când ele ajung la marginea exterioară sau la un obstacol. Noua configurație este notată cu $\mathbb{D}$. Intersecțiile segmentelor de linie din $\mathbb{D}$ definesc vârfurile grafului G, iar liniile care le interconectează definesc muchiile grafului.

Fiind dat un set de puncte Q, se poate obține un graf de căutare G_Q în $(\mathbb{R}, Q)$ după cum urmează. Se începe cu $\mathbb{D}$, căruia i se adaugă setul de puncte Q. Se extind apoi segmentele de linii orizontale și verticale de la fiecare punct din Q până când fiecare segment ajunge la un obstacol sau la margine. Noua configurație este notată $\mathbb{D}_Q$. Intersecțiile segmentelor de linie din $\mathbb{D}_Q$ definesc vârfurile grafului G_Q, iar segmentele de linie care le interconectează definesc muchiile grafului G_Q.

5.5.2.4 Rutarea globală orientată pe performanțe

Odată cu progresele înregistrate în tehnologia de fabricație a circuitelor VLSI, întârzierile datorate interconexiunilor au devenit semnificative pentru viteza circuitelor. De aceea, interconexiunile din cadrul unui circuit integrat și dintre circuite au un rol

major în determinarea performanøelor sistemelor digitale. Aceasta a determinat apariøia unor sisteme de proiectare în care accentul se pune pe maximizarea performanøelor. Deşi majoritatea cercetãrilor în acest domeniu au ca temã problema de plasare, existã unele lucrãri şi în domeniul rutării globale.

Cong *et al.* [60] au descris un algoritm de rutare globalã orientat pe performanøe pentru celule standard şi macro-celule. Aceastã metodã este bazatã arbori de rutare cu razã limitatã, pentru construcøia acestora utilizându-se euristici bazate pe algoritmul lui Prim pentru arbori de acoperire minimi. Metoda permite proiectantului un control al importanøei relative a costului rutării şi a întârzierilor de interconectare. Aceeaşi autori au elaborat o metodã care minimizeazã simultan lungimea totalã de interconectare (indicatã de costul arborelui) şi întârzierea maximã (indicatã de raza arborelui).

5.5.3 Rutarea globală prin metoda călirii simulate

Prima aplicare a metodei de călire simulatã pentru rutarea globalã a fost raportatã de Vecchi şi Kirkpatrick, autorii formulând problema ca o programare liniarã întreagã, unde capacitãøile fiecãrei muchii sunt egale cu unu [146]. Sunt considerate numai conexiuni cu douã terminale. Funcøia de cost utilizatã este suma pãtratelor încãrcãrilor tuturor regiunilor de rutare.

În aceastã secøiune, se descrie rutarea globalã utilizatã în pachetul de programe TimberWolf. Dupã faza de plasare iniøialã, programul TimberWolf continuã cu rutarea globalã. Rutarea globalã este soluøionatã în douã etape. Obiectivul primei etape este asignarea unor conexiuni canalelor de rutare orizontale astfel încât sã se minimizeze densitãøile canalelor. La sfârşitul acestei etape, sunt identificate toate conexiunile care pot fi asignate unui canal adiacent. Scopul etapei a doua este reducerea densitãøii canalelor prin modificarea asignării canalelor pentru conexiunile identificate anterior.

Dupã rutarea globalã, TimberWolf executã rafinarea plasării prin interschimbarea aleatoare a celulelor învecinate. Dupã fiecare interschimbare, se executã ambele etape ale rutării globale pentru rerutarea conexiunilor afectate de interschimbare. Metoda călirii simulate este utilizatã numai în a doua etapã a rutării globale, ca şi în faza de rafinare a plasării.

5.5.4 Rutarea globală prin metoda programării întregi

Rutarea globalã este formulatã ca o problemã de programare întreagã 0-1. Suprafaøa de rutare este modelatã ca un graf de caroiaj, unde fiecare nod reprezintã o celulã de caroiaj (super-celulã) [146]. Se presupune cã marginea dintre douã celule de caroiaj adiacente *l* şi *k* are o capacitate de $c_{l,k}$ piste. Aceasta corespunde unei ponderi pozitive $c_{l,k}$ a muchiei de legãturã dintre nodurile *l* şi *k* din graful de caroiaj. Pentru fiecare conexiune *i*, trebuie identificate modurile diferite de rutare ale conexiunii. Se presupune cã pentru fiecare conexiune *i* existã n_i arbori posibili de rutare $t_1^i, t_2^i, \ldots, t_{n_i}^i$.

Pentru un graf de caroiaj cu *M* muchii şi *T* arbori, arborii de rutare ai tuturor conexiunilor se pot reprezenta printr-o matrice 0-1 $A_{M\times T} = [a_{i,p}]$, unde:

$$a_{i,p} = \begin{cases} 1 & \text{dacă muchia } i \text{ aparţine arborelui } t_j^i \text{ şi lui } p, \text{ conform ecuaţiei (5.6)} \\ 0 & \text{în caz contrar;} \end{cases} \tag{5.5}$$

$$p = \sum_{m=1}^{l-1} n_m + k \,, \tag{5.6}$$

şi

$$T = \sum_{i=1}^{N} n_i \tag{5.7}$$

N fiind numărul de conexiuni.

Astfel, o formulare posibilă a rutării globale ca o problemă de programare întreagă este:

$$\begin{cases} \min \sum_{i=1}^{N} \sum_{j=1}^{n_i} g_{i,j} x_{i,j} & \text{cu condițiile:} \\ \sum_{j=1}^{n_i} x_{i,j} = 1 & 1 \le i \le N \\ \sum_{k=1}^{N} \sum_{l=1}^{n_k} a_{i,p} x_{l,k} \le c_i & 1 \le i \le M,\ p \text{ definit în (5.6)} \\ x_{k,j} = 0,1 & 1 \le k \le N,\ \text{şi } 1 \le j \le n_k \end{cases} \tag{5.10}$$

5.6 Rutarea detaliată

Rutarea detaliată este etapa următoare rutării globale. În această etapă se asignează fiecărei conexiuni piste particulare din cadrul regiunilor de rutare. Rutarea detaliată se poate împărți în rutare detaliată *generală* și rutare detaliată *restricționată*. În cazul rutării generale se impun un număr foarte redus de restricții asupra problemei de rutare, și este rutată o singură conexiune la un moment dat. Pe de altă parte, rutarea restricționată necesită anumite constrângeri asupra problemei de rutare, ca de exemplu zone rectangulare libere cu toți pinii aflați la periferie. Rutarea detaliată restricționată se poate împărți la rândul ei în rutare prin *canale* și rutare prin *blocuri de comutare*.

5.6.1 Rutarea detaliată generală

5.6.1.1 Rutarea labirint

O clasă de metode de rutare cu scop general care utilizează un model de grilă este reprezentată de rutarea labirint. Suprafața de rutare este împărțită în celule printr-o rețea de grile. Toate porturile și conexiunile sunt aliniate pe această grilă. Conectarea a două puncte se realizează prin determinarea unei secvențe de celule adiacente de la un punct la celălalt.

Unul din cei mai utilizați algoritmi pentru găsirea unei căi între două puncte pe o grilă cu obstacole este algoritmul introdus de Lee. O caracteristică a acestui algoritm este că dacă există o cale între două puncte, aceasta va fi găsită, și se garantează că aceasta este calea cea mai scurtă. Există trei faze în cadrul execuției algoritmului. Prima fază constă în etichetarea grilei, fiind numită faza de umplere sau de propagare a undelor. Perechea de celule care trebuie conectate este etichetată cu *S* și *T*. În această fază, în timpul pasului *i*, celulele libere aflate la distanța Manhattan *i* de la celula *S* sunt etichetate cu *i*. A doua fază a algoritmului este numită faza de căutare a sursei. Această procedură este inversa procedurii de umplere. În această fază este găsită calea cea mai scurtă. A treia fază este numită ștergerea etichetelor. În această fază etichetele tuturor celulelor, cu excepția celor utilizate pentru calea găsită, sunt șterse pentru interconexiunile următoare.

5.6.1.2 Rutarea prin metoda căutării liniilor

Unul din principalele dezavantaje ale algoritmului Lee și a variantelor sale este dimensiunea ridicată a memoriei necesare pentru reprezentarea suprafeței de rutare sub formă de grilă. Algoritmii de căutare a liniilor elimină acest dezavantaj.

Ideea principală a acestor algoritmi este următoarea. Se presupun două puncte *S* și *T* care trebuie conectate, și se consideră că inițial nu există obstacole pe suprafața de rutare. Dacă se trasează o linie verticală care trece prin *S* și o linie orizontală care

trece prin *T*, cele două linii se vor intersecta și vor indica o cale Manhattan între *S* și *T*. Dacă există obstacole, trebuie efectuate operații suplimentare pentru a găsi o cale între *S* și *T*.

Spre deosebire de algoritmul Lee, care efectuează o căutare în lățime, algorimii de căutare a liniilor efectuează o căutare în adâncime. De aceea, acești algoritmi nu garantează găsirea căii celei mai scurte, și pot necesita mai multe reveniri. În practică algoritmii de căutare a liniilor au o rată de completare a rutării similară cu algoritmul Lee, cu deosebirea că atât necesarul de memorie, cât și timpul de execuție este considerabil mai redus. Aceasta deoarece spațiul de rutare nu este memorat sub forma unei matrici, ci sub forma unor segmente de linii.

5.6.2 Rutarea prin canale

5.6.2.1 Definirea problemei de rutare prin canale

Un canal de rutare este definit printr-o regiune dreptunghiulară cu două rânduri de terminale de-a lungul marginilor de sus și de jos. Fiecărui terminal i se asociază un număr între 0 și *N*. Aceste numere reprezintă etichetele punctelor unei grile verticale, puncte aflate pe marginile de sus și de jos ale regiunii dreptunghiulare. Terminalele având aceeași etichetă *i* ($1 \leq i \leq N$) trebuie conectate prin interconexiunea *i*.

În general sunt disponibile două sau trei straturi pentru rutare. Segmentele de conexiuni orizontale numite *trunchiuri* sunt dispuse de-a lungul pistelor, iar segmentele de conexiuni verticale numite *ramuri* conectează trunchiurile la terminale.

Sarcina programului de rutare prin canale este de a specifica pentru fiecare conexiune un set de segmente de interconectare orizontale și verticale, ale căror puncte de sfârșit se află pe terminale sau pe pistele canalului. În cazul celulelor standard, obiectivul este de a se utiliza un număr minim de piste pentru efectuarea rutării complete. De aceea, numărul de piste necesare trebuie determinat de programul de rutare. În cazul rețelelor de porți obiectivul este de a se finaliza rutarea utilizând un număr specificat de piste.

5.6.2.2 Grafuri de constrângeri

Fiecărei probleme de rutare prin canale i se pot asocia două grafuri de constrângeri, dintre care una modelează constrângerile orizontale, iar cea de-a doua modelează constrângerile verticale. În cazul ambelor grafuri, fiecare conexiune este reprezentată printr-un vârf.

Graful constrângerilor orizontale GCH(*V*, *E*) este un graf nedirecționat în care un vârf $i \in V$ reprezintă conexiunea *i* și muchia $(i, j) \in E$ dacă segmentele orizontale ale conexiunii *i* și conexiunii *j* se suprapun.

Graful constrângerilor verticale GCV(*V*, *E*) este un graf direcționat în care fiecare nod $i \in V$ corespunde conexiunii *i*, și fiecare coloană verticală introduce o muchie $(i, j) \in E$ dacă și numai dacă conexiunea *i* are un pin în partea de sus a canalului și conexiunea *j* are un pin în partea de jos a canalului, în aceeași coloană.

5.6.2.3 Algoritmul marginii din stânga

Cele mai multe soluții ale problemei de rutare prin canale se bazează pe algoritmul marginii din stânga, existând diferite extensii și variații ale acestei tehnici. În această secțiune se va prezenta algoritmul de bază. Acest algoritm de rutare a fost propus de Hashimoto și Stevens. Algoritmul încearcă să maximizeze plasarea segmentelor orizontale în fiecare pistă. Segmentele conexiunilor sunt sortate în ordine

crescãtoare a distanøei marginii din stânga a acestora faøã de marginea din stânga a canalului. Algoritmul de bazã impune restricøia ca fiecare conexiune sã fie formatã dintr-un singur trunchi, şi ca trunchiurile sã fie rutate pe un strat, iar ramurile sã fie rutate pe celãlalt strat.

Procedura de asignare a trunchiurilor la segmente este urmãtoarea. Dupã sortarea conexiunilor în modul descris mai sus, algoritmul selectează trunchiul corespunzător primei conexiuni şi îl plasează în prima pistã de jos, eliminând apoi conexiunea din listã. Algoritmul parcurge apoi lista rãmasã pentru a gãsi prima conexiune care nu se suprapune cu conexiunea plasatã. Dacã o asemenea conexiune este gãsitã, ea este plasatã în aceeaşi pistã. Procesul este repetat pânã când nu se mai pot plasa conexiuni în prima pistã. Algoritmul continuã apoi cu plasarea conexiunilor în pista 2 şi apoi în celelalte piste, pânã când vor fi plasate toate conexiunile din listã.

5.6.2.4 Algoritmii Yoshimura şi Kuh

Dacã existã o cale n_1-n_2-n_3-…-n_k în graful constrângerilor verticale, atunci nu pot fi plasate douã conexiuni dintre {n_1, n_2, n_3, …, n_k} în aceeaşi pistã. Astfel, dacã cea mai lungã cale din punctul de vedere al numãrului de noduri este k, sunt necesare cel puøin k piste orizontale pentru realizarea interconexiunilor.

Yoshimura şi Kuh au elaborat doi algoritmi de rutare prin canale. Primul algoritm utilizează graful *GCV* şi reprezentarea prin zone a grafului *GCH*, încercând sã minimizeze calea cea mai lungã din *GCV*. Aceasta se realizează prin fuzionarea nodurilor din *GCV* (care corespund conexiunilor), astfel încât dupã fuzionare lungimea cãii celei mai lungi este minimizatã. Aceastã fuzionare este realizatã cu scopul de a minimiza densitatea canalelor. Al doilea algoritm propus de Yoshimura şi Kuh realizează minimizarea cãii celei mai lungi prin tehnici de combinare într-un graf bipartit.

5.7 Rutarea circuitelor FPGA

Rutarea circuitelor FPGA este mai dificilã decât rutarea clasicã, deoarece poziøia segmentelor utilizate pentru interconexiuni este fixã, iar interconectarea segmentelor este posibilã numai în locuri predefinite. La anumite arhitecturi FPGA cantitatea resurselor de rutare este redusã, ceea ce impune limitãri asupra numãrului alternativelor de rutare pentru o conexiune. În cazul circuitelor FPGA cu o conectivitate limitatã, rezolvarea conflictelor de rutare este esenøialã pentru obøinerea unei rutãri complete.

5.7.1 Problema de rutare a circuitelor FPGA

Metodele obişnuite de rutare globalã sau detaliatã nu sunt adecvate pentru circuitele FPGA. Din aceste motive, sunt necesare metode specifice pentru rutarea circuitelor FPGA.

Problema de rutare a circuitelor FPGA poate fi definitã astfel: Fiind datã o arhitecturã FPGA împreunã cu o configuraøie de asignare a pinilor blocurilor logice şi o colecøie de conexiuni între pini, sã se ruteze toate conexiunile fãrã a se depãşi resursele de rutare disponibile ale circuitului.

5.7.2 Rutarea prin expandarea grafului

Aceastã metodã de rutare detaliatã a fost elaboratã la Universitatea din Toronto, fiind implementatã în cadrul programului de rutare *CGE* (*Coarse Graph Expansion*) [27]. *CGE* presupune cã în prealabil a fost efectuatã rutarea globalã cu scopul de a echilibra densitatea canalelor de rutare. Rezultatele aratã cã programul *CGE* poate ruta circuite de dimensiuni relativ mari utilizând un numãr de piste apropiat de numã-

rul minim determinat de programul de rutare globalã. Programul de rutare *CGE* poate fi utilizat pentru diferite arhitecturi de circuite constând dintr-o reøea bidimensionalã de celule logice interconectate prin canale de rutare verticale øi orizontale.

Algoritmul de rutare *CGE* constã din douã faze. În *faza 1*, *CGE* expandeazã fiecare graf grosier øi memoreazã un subset al modurilor posibile în care conexiunea poate fi implementatã. Pentru fiecare graf $G(V, A)$, faza de expandare produce un graf expandat, numit $D(N, E)$. Muchiile acestui graf sunt etichetate cu un numãr care se referã la un segment specific de interconectare din circuitul FPGA. În *faza 2*, *CGE* plaseazã toate cãile de la toate grafurile expandate într-o singurã listã de cãi. Pe baza unei funcøii de cost, algoritmul selectează apoi cãi din această listã. Fiecare din acestea defineøte calea detaliatã a conexiunii corespunzãtoare.

5.7.3 Rutarea pe baza grafurilor cu ponderi multiple

Alexander øi Robins [2] au propus o abordare unitarã pentru rutarea circuitelor FPGA, care permite optimizarea simultanã a unor obiective multiple. Această abordare se bazează pe utilizarea unor grafuri multi-ponderate. Metoda combinã tehnici de rutare atât geometrice, cât øi combinatoriale, utilizând avantajele ambelor.

Pentru cele două categorii se pot alege diferite metode existente, rezultând o metodã hibridã. Pentru rutarea geometricã, în [2] s-a ales metoda de rutare 1-Steiner iteratã a lui Kahng øi Robins [99], iar pentru rutarea bazatã pe grafuri s-a ales metoda Kou, Markowsky øi Berman (*KMB*) de aproximare a arborilor Steiner. Metoda hibridã rezultatã este numitã algoritm 1-Steiner iterat pentru grafuri (*Graph Iterated 1-Steiner - GI1S*).

Metoda *GI1S* este în principiu o adaptare a metodei 1-Steiner iterate pentru grafuri. Însã, atunci când trebuie acoperit un subset N al nodurilor dintr-un graf, noøiunea arborelui de acoperire minim nu mai este bine definitã. În esenøã, un arbore de acoperire pentru N este acum un arbore Steiner, care nu mai poate fi construit într-un mod eficient, deoarece problema este NP-completã. Această dilemã poate fi rezolvatã prin înlocuirea construcøiei *MST* cu construcøia *KMB*. Astfel, fiind dat un graf $G = (V, E)$, conexiunea $N \subseteq V$, øi un set de puncte Steiner candidate, se poate defini reducerea costului ca fiind:

$$\Delta\overline{KMB}_G(N,S) = \overline{KMB}_G(N) - \overline{KMB}_G(N \cup S) \tag{5.13}$$

Algoritmul *GI1S* începe prin construirea arborelui *KMB*. Apoi, în fiecare iteraøie se determinã noduri Steiner candidate care reduc costul total *KMB*, noduri care sunt incluse în setul de noduri Steiner S. Costul arborelui *KMB* peste $N \cup S$ se va reduce cu fiecare nod adãugat, construcøia fiind terminatã atunci când nu mai există $x \in V$ cu $\Delta\overline{KMB}(N \cup S, \{x\}) > 0$.

5.8 Algoritm propus pentru rutarea circuitelor FPGA Atmel 6000

5.8.1 Arhitectura de rutare a circuitelor FPGA Atmel 6000

Arhitectura *Atmel* este formatã dintr-o reøea simetricã de celule identice. Reøeaua este continuã de la o margine a circuitului la alta, cu excepøia repetoarelor de magistralã care sunt amplasate la o distanøã de opt celule [7]. Magistralele permit o comunicaøie rapidã øi eficientã pe distanøe medii øi lungi. Există două tipuri de magistrale: magistrale locale øi magistrale expres.

Magistralele locale reprezintã legãtura dintre celule øi reøeaua de interconexiuni. Există două magistrale locale pentru fiecare coloanã de celule (NS1, NS2), øi două

magistrale locale pentru fiecare linie de celule (EW1, EW2). Într-un sector de opt celule fiecare magistralã localã este conectatã la fiecare celulã din coloana sau linia respectivã. În plus, fiecare celulã are posibilitatea de a ruta un semnal între magistralele NS1 şi EW1, sau între magistralele NS2 şi EW2 (întoarcere de 90°).

Magistralele expres nu sunt conectate direct la celule, asigurând astfel o vitezã mai ridicatã. Fiecãrei magistrale locale îi corespunde o magistralã expres, astfel încât existã douã magistrale expres pentru fiecare coloanã şi douã magistrale expres pentru fiecare linie de celule.

Fiecare magistralã localã şi expres este divizatã în segmente de cãtre unitãţi de conectare, numite *repetoare*, care sunt spaţiate la un interval de opt celule. Repetoarele sunt aliniate în linii şi coloane, partiţionând astfel reţeaua în sectoare de 8×8 celule.

5.8.2 Modelarea circuitului printr-un graf

Pentru utilizarea unor tehnici bazate pe grafuri pentru problema de rutare a circuitelor FPGA, circuitul trebuie modelat mai întâi ca un graf, astfel încât topologia grafului sã reflecte întreaga arhitecturã a circuitului. Cãile din acest graf corespund unor cãi de rutare fezabile din circuit, şi invers. Figura 5.1(a) prezintã o porţiune a arhitecturii circuitelor FPGA *Atmel*, iar Figura 5.1(b) ilustreazã graful de rutare corespunzãtor.

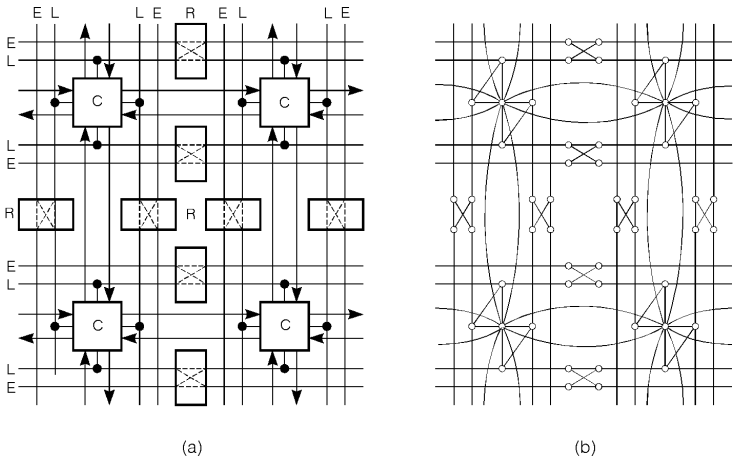

Figura 5.1. (a) Arhitectura circuitelor FPGA *Atmel*. (b) Graful de rutare corespunzãtor.

5.8.3 Descrierea algoritmului de rutare

Algoritmul de rutare implementat executã simultan rutarea globalã şi cea detaliatã. Un avantaj al acestei metode este cã estimarea preliminarã a rutării globale poate fi imediat corectatã. De asemenea, algoritmul poate lua în considerare efectele secundare pe care le au deciziile de rutare luate pentru o conexiune asupra celorlalte conexiuni, rezolvând astfel conflictele de rutare.

Algoritmul de rutare ţine cont de urmãtoarele aspecte, care sunt specifice pentru arhitectura FPGA *Atmel*:

1. Pentru rutarea semnalelor pe distanţe scurte, algoritmul utilizează celule libere în locul magistralelor. Magistralele sunt rezervate pentru rutarea semnalelor pe

distanøe mai mari, pentru semnale cu trei stări, sau pentru semnale cu un fan-out ridicat.

2. Algoritmul utilizează magistrale expres ori de câte ori este posibil.
3. Pentru creşterea performanøelor, algoritmul limitează numărul segmentelor magistralelor locale prin care trece un semnal şi utilizează repetoarele numai dacă este necesar. Se efectuează ramificarea semnalului de pe o magistrală expres la magistrala locală la fiecare repetor.

Fiecărui segment de interconectare al circuitului i se asociază două costuri: costul distanøei (C_d), care reflectă întârzierile de rutare asociate cu segmentul de interconectare, şi costul competiøiei (C_c), care contorizează legăturile care se află în competiøie pentru acelaşi segment. Fiecărei căi din lista de conexiuni i se asociază de asemenea două costuri: suma costului distanøelor (S_d) pentru segmentele căii şi suma costului competiøiei (S_c) pentru segmentele căii.

Algoritmul nu poate considera toate posibilităøile de interconectare din cadrul circuitului într-o singură etapă, în scopul reducerii necesarului de memorie şi al timpului de execuøie. Acesta este motivul pentru care se utilizează o metodă iterativă. În prima etapă se consideră numai acele căi posibile pentru o conexiune care corespund costului C_d minim. Dacă aceste căi sunt în conflict cu conexiunile deja rutate, algoritmul continuă căutarea pornind de la costul căii eşuate.

În prima etapă, se construieşte graful de conectivitate pe baza structurii circuitului FPGA, iar apoi se construieşte graful de rutare pe baza rezultatelor obøinute după maparea tehnologică şi plasare. Prin legături directe se desemnează acele legături care nu implică utilizarea unei magistrale.

În următoarea etapă, algoritmul încearcă divizarea conexiunilor în legături directe în cazul în care acestea nu trebuie să treacă prin mai mult de cinci celule. Pentru fiecare conexiune, algoritmul determină distanøa minimă a căii de rutare, şi memorează toate alternativele de rutare într-o listă a conexiunilor.

Algoritmul poate efectua două tipuri de optimizări: din punct de vedere al spaøiului ocupat şi din punct de vedere al vitezei. Pentru optimizarea din punctul de vedere al *spaøiului*, algoritmul sortează mai întâi conexiunile după numărul alternativelor posibile de rutare (numărul căilor din graf, utilizând costul S_c), astfel încât conexiunile care au un număr mai redus de alternative vor fi mai prioritare. Pentru optimizarea din punctul de vedere al *vitezei*, algoritmul sortează mai întâi conexiunile după lungimea lor (utilizând costul S_d). Astfel, va determina ca liniile lungi să aleagă magistralele expres, care sunt mai rapide. Dintre toate conexiunile posibile, se alege cea cu costul S_c minim.

Funcøia de cost utilizată în cazul optimizării pentru rutabilitate are rolul de a selecta o cale de rutare care va avea un efect negativ redus asupra conexiunilor rămase, din punct de vedere al rutabilităøii. Această funcøie împiedică selectarea căilor care conøin segmente de interconectare pentru care există un număr mare de cereri.

5.9 Concluzii

În acest capitol a fost prezentată problema de rutare a circuitelor VLSI şi FPGA, şi a fost propus un algoritm de rutare pentru circuitele FPGA cu resurse limitate de rutare, în particular pentru circuitul FPGA *Atmel*. Rutarea se descompune de obicei în două etape: *rutarea globală* şi *rutarea detaliată*.

Există diferite metode de *rutare globală*: metode secvenøiale, metode aleatoare, metoda programării liniare şi metoda descompunerii ierarhice. A fost prezentată metoda parcurgerii labirintului, o metodă de rutare globală orientată pe performanøe bazată arbori de rutare cu rază limitată, metoda de călire simulată, metoda programării întregi.

Au fost prezentate două metode de *rutare detaliată* generală: metoda parcurgerii labirintului și metoda căutării liniilor. Algoritmii de parcurgere a labirintului garantează găsirea căii celei mai scurte dacă o asemenea cale există. Din această categorie a fost descris algoritmul Lee, care necesită însă un timp de execuție ridicat și un necesar ridicat de memorie. Algoritmii de căutare a liniilor elimină aceste dezavantaje ale algoritmului Lee. Acești algoritmi garantează găsirea unei căi dacă o asemenea cale există, nu neapărat cea mai scurtă.

Dintre metodele de rutare prin canale, a fost descris algoritmul marginii din stânga și doi algoritmi elaborați de Yoshimura și Kuh. Primul algoritm utilizează fuzionarea nodurilor astfel încât după fuzionare lungimea căii celei mai lungi este minimizată. Al doilea algoritm minimizează lungimea căii celei mai lungi prin tehnici de potrivire într-un graf bipartit.

A fost definită problema de rutare pentru circuitele FPGA, punându-se în evidență modul în care această problemă este diferită de problema de rutare a circuitelor VLSI. În literatură a fost publicat un număr redus de programe de rutare pentru circuitele FPGA. Au fost prezentate două din acestea: rutarea prin expandarea grafului și rutarea bazată pe grafuri cu ponderi multiple.

În acest capitol s-a descris un algoritm de rutare care a fost conceput și implementat pentru circuitele FPGA *Atmel*. Algoritmul execută simultan rutarea globală și cea detaliată. Un avantaj al acestei metode este că estimarea preliminară a rutării globale poate fi imediat corectată. De asemenea, algoritmul poate lua în considerare efectele secundare pe care le au deciziile de rutare luate pentru o conexiune asupra celorlalte conexiuni, rezolvând astfel conflictele de rutare. Algoritmul poate efectua două tipuri de optimizări: din punct de vedere al spațiului ocupat și din punct de vedere al vitezei.

6. SISTEM CAD PENTRU PROIECTAREA SISTEMELOR NUMERICE CU CIRCUITELE FPGA ATMEL

6.1 Structura generală a sistemului CAD

Structura sistemului CAD care a fost implementat pentru proiectarea cu circuitele FPGA *Atmel* este prezentată în Figura 6.1.

Intrarea pentru sistemul CAD o constituie descrierea funcțională a sistemului numeric în limbajul *ABEL*. Această descriere este compilată prin apelarea modulului corespunzător al sistemului de dezvoltare *Easy-ABEL*. Fișierul generat, în formatul PLA, este optimizat cu ajutorul modulului *PLAOpt*, care minimizează logica astfel încât vor fi utilizați mai puțini termeni produs în circuitul utilizat pentru implementare. Se pot introduce diferite opțiuni de optimizare și de minimizare logică. În urma optimizării se obține de asemenea un fișier în formatul PLA. Acest fișier este translatat apoi într-un fișier de tip PDS, care conține ecuațiile sistemului numeric.

Modulele implementate ale sistemului CAD pornesc de la fișierul PDS. Pe baza acestuia se generează reprezentarea internă a sistemului sub forma unui graf. Pentru transformarea ecuațiilor într-o formă corespunzătoare circuitului FPGA, se execută etapa de mapare tehnologică. Etapa următoare este cea de plasare, în care se selectează locații specifice pentru fiecare celulă logică. Modulul de rutare efectuează alocarea resurselor de rutare în scopul interconectării celulelor logice, generându-se un fișier de configurare al circuitului. În final, structura circuitului configurat poate fi vizualizată în mod grafic.

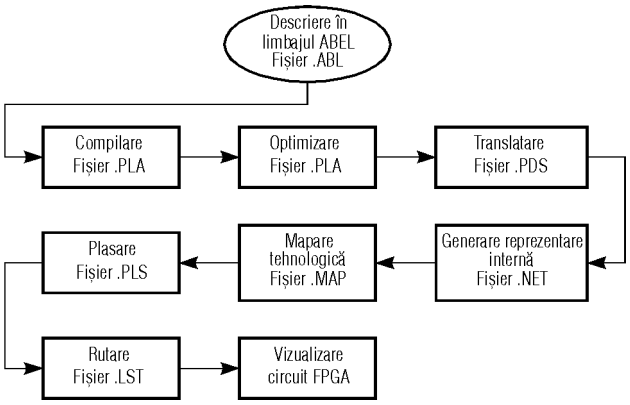

Figura 6.1. Structura generalã a sistemului CAD.

6.2 Descrierea proiectului

Descrierea proiectului se realizeazã în limbajul de descriere *ABEL-HDL* (*ABEL Hardware Description Language*). Acest limbaj a fost elaborat de firma *Data I/O*, pe baza limbajului de proiectare *ABEL*, reprezentând un standard industrial. Permite descrieri funcøionale de nivel înalt, ca ecuaøii, tabele de adevãr, diagrame de stare.

Descrierile pot fi introduse fãrã a øine cont de arhitectura circuitului care va fi utilizat pentru implementare. Descrierile independente de arhitecturã (care nu conøin declaraøii de dispozitive şi declaraøii de numere de pini) trebuie sã fie mai cuprinzãtoare decât cele specifice pentru o anumitã arhitecturã.

6.3 Compilarea şi optimizarea descrierilor

Compilarea este realizatã prin apelarea modulului *AHDL2PLA*, care converteşte diagramele de stare şi tabelele de adevãr în ecuaøii booleene, translateazã vectorii de test, expandeazã macrourile şi verificã sintaxa descrierii. Compilatorul realizeazã sinteza sistemului şi generează un fişier în formatul PLA şi un fişier cu vectorii de test. Dacã apar erori, tipul acestora şi locul în care apar sunt scrise într-un fişier listing (*.lst).

Optimizarea ecuaøiilor se realizeazã cu ajutorul modulului *PLAOpt*, care minimizează logica astfel încât vor fi utilizaøi mai puøini termeni produs în circuitul utilizat pentru implementare. Ecuaøiile care vor fi minimizate sunt citite din fişierul PLA, iar rezultatele sunt scrise în fişierul de ieşire, de asemenea în formatul PLA.

6.4 Generarea reprezentării interne

Din fişierul sursã conøinând descrierea de intrare se generează o descriere intermediarã utilizând compilatorul sistemului de dezvoltare *Easy-ABEL*. Această descriere este în formatul unui fişier .PDS, conøinând ecuaøiile sistemului descris. Fiecare ecuaøie din fişierul .PDS este analizatã şi se generează un graf, ale cãrui noduri sunt componente logice de bazã.

După construirea grafului fiecãrei ecuaøii, noul graf trebuie adãugat grafului general al circuitului. O etapã importantã este eliminarea redundanøelor. În locul unor grafuri separate multiple, aceste grafuri sunt combinate într-unul singur, dacã este po-

sibil, încercându-se minimizarea numărului de noduri, și în consecinøã, complexitatea circuitului.

6.5 Maparea tehnologică

Maparea tehnologicã este operaøia de transformare a unei reprezentări logice cu nivele multiple într-o interconexiune de elemente logice dintr-o bibliotecã datã de elemente. Aceastã operaøie este o etapã importantã a sintezei circuitelor logice pentru diferite tehnologii, ca reøele de porøi, celule standard, sau circuite FPGA. Calitatea circuitelor sintetizate depinde în mare mãsurã de aceastã etapã.

Pentru sistemul CAD propus, maparea tehnologicã implementatã se bazează pe o metodã algoritmicã. Algoritmul de mapare tehnologicã utilizat pornește de la reprezentarea internã a circuitului. Primele etape ale mapării tehnologice sunt partiøionarea și decompoziøia. Aceste etape au fost implementate în cadrul etapei de generare a reprezentării interne, reøeaua logicã generatã având proprietãøile cerute. Reøeaua logicã este deja descompusã, conøinând numai componentele logice de bazã [16]. De aceea, singura etapã care trebuie implementatã este acoperirea, care include și etapa de potrivire booleanã. Pentru acoperirea reøelei, trebuie sã se ia în considerare setul de configuraøii logice disponibile în cadrul circuitului FPGA *Atmel* 6002.

6.6 Plasarea celulelor

Pentru plasare s-a implementat un algoritm care utilizează partiøionarea pe baza tãieturii minime. Algoritmul de plasare implementat utilizează bipartiøionarea a cãrei funcøie obiectiv urmărește minimizarea lungimii interconexiunilor simultan cu minimizarea diferenøei între numărul de conexiuni din cele două porøiuni. Această diferenøã este mãsura care urmărește distribuøia echilibratã a conexiunilor din cele două porøiuni. În cadrul algoritmului de plasare se aplicã în mod recursiv bipartiøionarea până când se ajunge la porøiuni care conøin o singurã celulã a circuitului. Liniile de tãieturã se aplicã în mod alternativ, pe orizontalã și pe verticalã.

6.7 Rutarea circuitului

Algoritmul de rutare implementat executã simultan rutarea globalã și cea detaliatã. Se eliminã astfel dezavantajul legat de împãrøirea problemei de rutare în două subprobleme, cea de rutare globalã și de rutare detaliatã, care conduce la rezultate globale care nu sunt optime. Un alt avantaj al acestei metode este cã estimarea preliminarã a rutării globale poate fi imediat corectatã. De asemenea, algoritmul poate lua în considerare efectele secundare pe care le au deciziile de rutare luate pentru o conexiune asupra celorlalte conexiuni.

Pentru rutarea semnalelor pe distanøe scurte, algoritmul utilizează celule libere în locul magistralelor. Magistralele sunt rezervate pentru rutarea semnalelor pe distanøe mai mari, pentru semnale cu trei stări, sau pentru semnale cu un fan-out ridicat.

6.8 Concluzii

În acest capitol s-a prezentat un sistem CAD conceput și implementat pentru proiectarea sistemelor numerice utilizând circuitele FPGA din seria *Atmel* 6000. Sistemul CAD a fost implementat în limbajul C++, funcøionând sub sistemul de operare Windows 95. Sistemul a fost conceput pe baza metodologiei de proiectare care a fost propusã în cadrul tezei, și integrează rezultatele obøinute în studiul etapelor de partiøionare, plasare și rutare. În plus, sistemul realizează maparea tehnologicã pentru circuitul FPGA utilizat. A fost realizatã de asemenea o interfaøã cu sistemul de proiectare

EasyABEL, interfaøã care permite descrierea sistemului numeric în limbajul *ABEL*, compilarea descrierii într-un set de ecuaøii şi optimizarea acestora.

7. CONCLUZII FINALE

Obiectivul urmãrit pe parcursul tezei l-a constituit studiul etapelor de proiectare fizicã *pentru circuite FPGA cu resurse limitate de rutare*. Principala problemã pentru aceste circuite o reprezintã asigurarea rutabilitãøii, şi îndeplinirea acestui obiectiv este urmãritã în trei etape importante ale procesului de proiectare fizicã: *partiøionare*, *plasare* şi *rutare*. Dacã circuitele FPGA cu arhitecturi de rutare bazate pe canale segmentate, ca de exemplu circuitele FPGA Xilinx, sunt studiate în mãsurã mai mare în literatura de specialitate, circuitele cu resurse limitate de rutare au fost studiate într-o mãsurã foarte redusã. Teza aduce contribuøii în acest domeniu.

Contribuøiile teoretice ale tezei sunt urmãtoarele:

1. Elaborarea unei metodologii de proiectare asistatã de calculator a sistemelor numerice pentru circuite FPGA cu resurse limitate de rutare. În metodologia propusã, asigurarea rutabilitãøii circuitului este urmãritã nu numai în cadrul etapei de rutare propriu-zisã, ci încã din etapa de partiøionare, continuând cu etapa de plasare şi apoi cea de rutare.
2. Propunerea unei metrici mai adecvate pentru partiøionarea utilizatã la plasarea circuitelor FPGA cu resurse limitate de rutare.
3. Elaborarea unui algoritm de partiøionare care urmãreşte echilibrarea numãrului de conexiuni din cadrul partiøiilor, minimizând în acelaşi timp lungimea totalã a interconexiunilor.
4. Elaborarea unui algoritm genetic pentru partiøionare, cu un timp de execuøie mai redus decât cel al unui algoritm bazat pe metoda cãlirii simulate, calitatea soluøiilor obøinute fiind comparabilã cu cea obøinutã prin metoda cãlirii simulate.
5. Elaborarea şi testarea unui algoritm de plasare pe baza bipartiøionãrii, a cãrei funcøie obiectiv urmãreşte minimizarea lungimii interconexiunilor simultan cu distribuøia echilibratã a conexiunilor din cele douã porøiuni.
6. Propunerea unei secvenøe de aplicare a liniilor de tãieturã care este mai eficientã decât secvenøele tradiøionale, având ca efect o reducere a dimensiunii maxime a tãieturii, cât şi a sumei totale a tãieturilor, deci implicit a lungimii totale a interconexiunilor.
7. Elaborarea unui algoritm genetic pentru plasarea circuitelor FPGA, având ca obiectiv atât reducerea lungimii totale a interconexiunilor, cât şi asigurarea rutabilitãøii circuitului.
8. Elaborarea unui algoritm de rutare pentru circuitele FPGA *Atmel*, având urmãtoarele avantaje: executarea simultanã a rutãrii globale şi a celei detaliate, considerarea efectelor secundare pe care le au deciziile de rutare luate pentru o conexiune asupra celorlalte conexiuni, optimizarea atât din punct de vedere al spaøiului ocupat, cât şi din punct de vedere al vitezei.

Contribuøiile practice ale tezei sunt urmãtoarele:

1. Implementarea şi testarea algoritmilor de partiøionare, plasare şi rutare pentru circuitele FPGA cu resurse limitate de rutare.
2. Implementarea unui sistem CAD pentru proiectarea sistemelor numerice utilizând circuitele FPGA din seria *Atmel* 6000, sistem care se bazeazã pe me-

todologia de proiectare propusã în cadrul tezei, şi integrează rezultatele obţinute în studiul etapelor de partiţionare, plasare şi rutare. Sistemul conţine în plus un program de mapare tehnologicã pentru circuitele FPGA *Atmel* şi o interfaţã care permite utilizarea limbajului de descriere *ABEL*.

Algoritmii elaboraţi pentru etapele de proiectare fizicã a circuitelor FPGA, ca şi sistemul CAD implementat, pot fi îmbunãtãţiţi în mai multe moduri. De asemenea, pot fi investigate şi alte metode de soluţionare a problemelor de proiectare fizicã pentru circuitele FPGA. Unele din dezvoltãrile posibile sunt urmãtoarele.

1. Modificarea algoritmului genetic pentru partiţionare şi al celui pentru plasare astfel încât principalii parametrii sã nu fie definiţi în mod static, ci aceştia sã treacã printr-un proces de optimizare pe parcursul execuţiei algoritmului.
2. Investigarea posibilitãţii de utilizare a algoritmilor genetici pentru problema de rutare a circuitelor FPGA, şi testarea eficienţei unui asemenea algoritm.
3. Extinderea sistemului CAD astfel încât sã fie posibilã utilizarea şi a altor tipuri de circuite FPGA.
4. Extinderea sistemului CAD cu o interfaţã pentru limbajul de descriere VHDL sau Verilog.
5. Implementarea paralelã a algoritmilor genetici pentru plasare şi a algoritmului de plasare prin metoda cãlirii simulate.
6. Implementarea paralelã a algoritmului de rutare.

BIBLIOGRAFIE (selectivă)

[2] Alexander, M. J., Robins, G., “An Architecture-Independent Unified Approach to FPGA Routing”, Technical Report No. CS-93-51, Department of Computer Science, University of Virginia, Charlottesville, 1993.

[4] Alexander, M. J., Robins, G., “New Performance-Driven FPGA Routing Algorithms”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 15, No. 12, December 1996, pp. 1505-1517.

[6] Altera Corp., *The Maximalist Handbook*, 1990.

[7] Atmel Corp., *Configurable Logic. PLD, FPGA, Gate Array Data Book*, San Jose, 1995.

[8] Baruch, Z., “Datapath Allocation”, *ACAM Scientific Journal*, Vol. 4, No. 2, 1995, pp. 67-75.

[9] Baruch, Z., “Metode de descriere a sistemelor numerice”. Referat de doctorat, Catedra de Calculatoare, Universitatea Tehnicã din Cluj-Napoca, 1995.

[10] Baruch, Z., “Scheduling Algorithms for High-Level Synthesis”, *ACAM Scientific Journal*, Vol. 5, No. 1-2, 1996, pp. 48-57.

[11] Baruch, Z., “Sistem CAD pentru sinteza sistemelor numerice”. Referat de doctorat, Catedra de Calculatoare, Universitatea Tehnicã din Cluj-Napoca, 1996.

[12] Baruch, Z., “Translatarea limbajelor de descriere a unitãţilor hardware”. Referat de doctorat, Catedra de Calculatoare, Universitatea Tehnicã din Cluj-Napoca, 1996.

[13] Baruch, Z., Creţ, O., Pusztai, K., “CAD System for the Atmel FPGA Circuits”. Third International Conference on Technical Informatics CONTI98, Timişoara, 1998, In *Transactions on Automatic Control and Computer Science*, Vol. 43 (57), No. 4, pp. 228-237.

[14] Baruch, Z., Creţ, O., Pusztai, K., “Partitioning for FPGA Circuits”. In *Proceedings of Micro-CAD '97 International Computer Science Meeting*, Miskolc, Hungary, 1997, Section D, pp. 113-116.

[15] Baruch, Z., Creţ, O., Pusztai, K., “Routing for FPGA Circuits”. In *Proceedings of A&Q ’98 International Conference on Automation and Quality Control*, Cluj-Napoca, 1998, pp. Q214-Q219.

[16] Baruch, Z., Creţ, O., Pusztai, K., “Technology Mapping for the Atmel FPGA Circuits”. Third International Conference on Technical Informatics CONTI98, Timişoara, 1998, In *Transactions on Automatic Control and Computer Science*, Vol. 43 (57), No. 4, pp. 218-227.

[17] Baruch, Z., Pusztai, K., “AHPL Compiler”. In *Proceedings of MicroCAD '93 International Computer Science Meeting*, Miskolc, Hungary, 1993, Section E, pp. 121-129.

[23] Brasen, D., Saucier, G., “FPGA Partitioning for critical paths”. In *The European Design and Test Conference*, IEEE, 1994, pp. 99-103.

[26] Brown, S., “FPGA Architectural Research: A Survey”, *IEEE Design & Test of Computers*, Winter 1996, pp. 9-15.

[27] Brown, S., *Routing Algorithms and Architectures for Field-Programmable Gate Arrays*, PhD Thesis, Department of Electrical and Computer Engineering, University of Toronto, Canada, 1992.

[31] Bui, T. N., Moon, B. R., “Genetic Algorithm and Graph Partitioning”, *IEEE Transactions on Computers*, Vol. 45, No. 7, July 1996, pp. 841-855.

[34] Cai, H., Otten, R. J. H. M., “Conflict-Free Channel Definition in Building-Block Layout”. *IEEE Transactions on Computer-Aided Design*, Vol. CAD-8, No. 9, September 1989, pp. 838-847.

[39] Chan, P. K., Schlag, M. D. F., Zien, J. Y., “Spectral K-Way Ratio Cut Partitioning and Clustering”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 13, No. 9, September 1994, pp. 1088-1096.

[40] Chan, P. K., Schlag, M. D. F., Zien, J. Y., “Spectral-Based Multiway FPGA Partitioning”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 15, No. 5, May 1996, pp. 554-560.

[48] Chen, C. D., Lee, Y. S., Wu, A. C. H., Lin, Y. L., “TRACER-fpga: A Router for RAM-Based FPGA’s”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 14, No. 3, March 1995, pp. 371-374.

[50] Cheng, C. K., Wei, Y. C., “An Improved Two-Way Partitioning Algorithm with Stable Performance”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 10, No. 12, December 1991, pp. 1502-1511.

[53] Chiang., C., Sarrafzadeh, M., Wong, C. K., “Global Routing Based on Steiner Min-Max Trees”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 9, No. 12, December 1990, pp. 1318-1325.

[54] Chiang, C., Wong, C. K., Sarrafzadeh, M., “A Weighted Steiner Tree-Based Global Router with Simultaneous Length and Density Minimization”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 13, No. 12, December 1994, pp. 1461-1469.

[60] Cong, J., Kahng, A. B., Robins, G., Sarrafzadeh, M., Wong, C. K., “Provably Good Performance-Driven Global Routing”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 11, No. 6, June 1992, pp. 739-752.

[62] Cormen, T. H., Leiserson, C., Rivest, R., *Introduction to Algorithms*, The MIT Press/ McGraw-Hill Book Company, 1991.

[69] Dutt, S., Deng, W., “A Probability-Based Approach to VLSI Circuit Partitioning”, In *Proceedings of the 33rd Design Automation Conference*, ACM/IEEE, 1996, pp. 100-105.

[72] Eberle, H., Gehring, S., Ludwig, S., Wirth, N., "Tools for Digital Circuit Design Using FPGAs". Technical Report No. 215, Institute for Computer Systems, ETH Zürich, May 1994.

[73] Esbensen, H., “A Genetic Algorithm for Macro Cell Placement”, in *Proceedings of the European Design Automation Conference*, 1992, pp. 52-57.

[79] Gajski, D. D., Dutt, N. D., Wu, C. H., Lin, Y. L., *Introduction to Chip and System Design*. Kluwer Academic Publishers, 1992.

[86] Hagen, L., Kahng, A. B., "New Spectral Methods for Ratio Cut Partitioning and Clustering", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 11, No. 9, September 1992, pp. 1074-1085.

[89] Hasan, Z., Harrison, D., Ciesielski, M., "A Fast Partitioning Method for PLA-Based FPGAs", *IEEE Design & Test of Computers*, Vol. 9, December 1992, pp. 34-39.

[98] Johannes, F. M., "Partitioning of VLSI Circuits and Systems". In *Proceedings of the 33rd Design Automation Conference*, ACM/IEEE, 1996, pp. 83-87.

[99] Kahng, A. B., Robins, G., "A New Class of Iterative Steiner Tree Heuristics with Good Performance", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 11, No. 7, July 1992, pp. 893-902.

[101] Kim, C., Shin, H., "A Performance-Driven Logic Emulation System: FPGA Network Design and Performance-Driven Partitioning", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 15, No. 5, May 1996, pp. 560-568.

[104] Kravitz, S. A., Rutenbar, R. A., "Placement by Simulated Annealing on a Multiprocessor", *IEEE Transactions on Computer-Aided Design*, Vol. CAD-6, No. 4, July 1987, pp. 534-549.

[105] Krishnamurthy, B., "An Improved Min-Cut Algorithm for Partitioning VLSI Networks", *IEEE Transactions on Computers*, Vol. 33, No. 5, May 1984, pp. 438-446.

[109] Li, J., Lillis, J., Liu, L. T., Cheng, C. K., "New Spectral Linear Placement and Clustering Approach", In *Proceedings of the 33rd Design Automation Conference*, ACM/IEEE, 1996, pp. 88-93.

[121] Michel, P., Lauther, U., Duzy, P., *The Synthesis Approach to Digital System Design*. Kluwer Academic Publishers, 1992.

[122] De Micheli, G., *Synthesis and Optimization of Digital Circuits*. McGraw-Hill, 1994.

[123] Mîndru, F., Baruch, Z., Mîndru, A., "Global Router for an Array-Based Model of FPGAs", *Acta Electrotehnica Napocensis*, Mediamira Science Publisher, Vol. 17, No. 1, pp. 63-66.

[124] Mohan, S., Mazumder, P., "Wolverines: Standard Cell Placement on a Network of Workstations", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 12, No. 9, September 1993, pp. 1312-1326.

[127] Nedevschi, S., Pusztai, K., Baruch, Z., Creø, O., "Introducerea tehnologiei FPGA în învăøământul, cercetarea şi industria românească". Raport de cercetare pentru Contractul Nr. 8003/98, Tema Nr. 54, Universitatea Tehnicã din Cluj-Napoca, 1998.

[129] Oommen, B. J., St. Croix, E. V., "Graph Partitioning Using Learning Automata", *IEEE Transactions on Computers*, Vol. 45, No. 2, February 1996, pp. 195-208.

[133] Pusztai, K., Baruch, Z., Creø, O., "Configurarea automatã a circuitelor FPGA". Raport de cercetare pentru Contractul Nr. 5003/95, Tema Nr. 124, Universitatea Tehnicã din Cluj-Napoca, 1996.

[134] Pusztai, K., Baruch, Z., Balint, P., Harsanyi, A., "Interfaøã cu sisteme de proiectare a circuitelor". Raport de cercetare pentru Contractul Nr. 4003/94, Tema Nr. B29, Universitatea Tehnicã din Cluj-Napoca, 1995.

[135] Pusztai, K., Baruch, Z., Creø, O., "Sistem CAD pentru proiectarea cu circuite FPGA". Raport de cercetare pentru Contractul Nr. 7003/97, Tema Nr. 32, Universitatea Tehnicã din Cluj-Napoca, 1997.

[141] Rose, J., Snelgrove, W., Vranesic, Z., "Parallel Standard Cell Placement Algorithms with Quality Equivalent to Simulated Annealing", *IEEE Transactions on Computer-Aided Design*, Vol. CAD-7, No. 3, March 1988, pp. 387-396.

[142] Roussel-Ragot, P., Dreyfus, G., "A Problem Independent Parallel Implementation of Simulated Annealing: Models and Experiments", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 9, No. 8, August 1990, pp. 827-835.

[146] Sait, S. M., Youssef, H., *VLSI Physical Design Automation*, McGraw-Hill Book Company, 1995.

[150] Schnecke, V., Vornberger, O., "Genetic Design of VLSI-Layouts", in *Proceedings of 1st IEE/IEEE International Conference on GAs in Engineering Systems: Innovations and Applications*, GALESIA '95, Sheffield, U.K., Sept. 1995, pp. 430-435.

[154] Shih, P., Chang, K., Feng, W., "Neural Computation Network for Global Routing". *Computer Aided Design*, Vol. 23, No. 8, October 1991, pp. 539-547.

[156] Shin, H., Kim, C., "A Simple Yet Effective Technique for Partitioning", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 1, No. 3, September 1993, pp. 380-386.

[159] Sun, Y., Wang, T. C., Wong, C. K., Liu, C. L., "Routing for Symmetric FPGA's and FPIC's", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 16, No. 1, January 1997, pp. 20-31.

[167] Togawa, N., Sato, M., Ohtsuki, T., "A Simultaneous Placement and Global Routing Algorithm for Symmetric FPGAs", *The 2nd International ACM/SIGDA Workshop on Field-Programmable Gate Arrays*, Berkeley, CA, 1994.

[172] Vannelli, A., "An Adaptation of the Interior Point Method for Solving the Global Routing Problem", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 10, No. 2, February 1991, pp. 193-203.

[175] Wei, Y. C., Cheng, C. K., "Ratio Cut Partitioning for Hierarchical Designs", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 10, No. 7, July 1991, pp. 911-921.

[177] Xilinx Inc., *The Programmable Gate Array Data Book*, San Jose, 1991.

[178] Xilinx Inc., *XACT Development System Reference Guide*, San Jose, 1993.

[179] Yang, H. H., Wong, D. F., "Circuit Clustering for Delay Minimization under Area and Pin Constraints", Technical Report No. TR-94-03, Department of Computer Sciences, University of Texas at Austin, Austin, 1994.

[180] Yang, H. H., Wong, D. F., "Efficient Network Flow Based Min-Cut Balanced Partitioning", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 15, No. 12, December 1996, pp. 1533-1540.

[183] Yeh, C. W., Cheng, C. K., Lin, T. T. Y., "A General Purpose, Multiple-Way Partitioning Algorithm", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 13, No. 12, December 1994, pp. 1480-1488.

[186] Yeh, C. W., Cheng, C. K., Lin, T. T. Y., "Optimization by Iterative Improvement: An Experimental Evaluation on Two-Way Partitioning", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 14, No. 2, February 1995, pp. 145-153.

[188] Zhu, K., Wong, D. F., "A Timing-Driven Global Router for Symmetrical Array Based FPGAs", Technical Report No. TR-94-22, Department of Computer Sciences, University of Texas at Austin, Austin, 1994.

[189] Zobrist, G. W. (Editor), *Routing, Placement, and Partitioning*, Ablex Publishing Corporation, Norwood, New Jersey, 1994.

[190] Zurada, J. M., *Introduction to Artificial Neural Systems*, West Publishing Company, 1992.