4. PLASAREA MODULELOR CU OBIECTIVUL ASIGURĂRII RUTABILITĂȚII CIRCUITELOR

4.1 Introducere

Plasarea este procesul de aranjare a componentelor unui circuit pe o suprafaøã. În cazul circuitelor FPGA, plasarea semnificã asignarea funcøiilor logice diferitelor celule ale circuitului, a cãror poziøie este fixã. Plasarea este o etapã importantã a procesului de proiectare, deoarece în aceastã etapã 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.

În proiectarea circuitelor, existã diferite funcøii obiectiv care trebuie minimizate, ca de exemplu lungimea conexiunilor, suprafaøa ocupatã, puterea disipatã, sau numãrul conexiunilor care se intersectează. Practic, este imposibilã minimizarea simultanã a tuturor acestor obiective, fiind necesarã definirea unui set de criterii care reflectã aceste obiective. De obicei, este importantã minimizarea lungimii totale de rutare. Totuşi, existã şi alte obiective care pot fi critice. De exemplu, în etapa de rutare, dacã problema este rutabilitatea circuitului, obiectivul poate fi modificat pentru a reflecta congestia conexiunilor în locul lungimii totale a acestora.

Atunci când este mãsuratã distanøa între douã module, existã douã metrici care pot fi utilizate: distanøa Manhattan sau cea euclidianã. De exemplu, dacã poziøiile a douã module pe o placã sunt (x_1, y_1), respectiv (x_2, y_2), distanøa Manhattan este definitã prin:

$$d = |x_1 - x_2| + |y_1 - y_2| \tag{4.1}$$

iar distanøa euclidianã este definitã prin:

$$d = (x_1 - x_2)^2 + (y_1 - y_2)^2 \tag{4.2}$$

Ca un exemplu, se considerã circuitul din Figura 4.1(a). Se presupune cã este necesarã plasarea porøilor pe o suprafaøã bidimensionalã. În Figura 4.1(b) este ilustratã o asemenea plasare, iar în Figura 4.1(c) este ilustratã aceeaşi plasare într-o formã simbolicã. În cazul plasării simbolice, detaliile de rutare sunt omise. Din această plasare simbolicã se poate obøine însã o estimare a cerinøelor de rutare. Considerând Figura 4.1(c), presupunem cã rutarea unei conexiuni de la un modul la altul necesitã o lungime a conexiunilor egalã cu distanøa Manhattan între module. De exemplu, lungimea conexiunii (3,6) este de 2 unitãøi, lungimea totalã a conexiunilor pentru plasarea din Figura 4.1(c) fiind de 10 unitãøi. În Figura 4.1(d) se prezintã o altă plasare simbolicã pentru care lungimea totalã a conexiunilor ω este de 12. Figura 4.1(e) ilustrează o plasare unidimensionalã a circuitului, care necesitã de asemenea o lungime a conexiunilor de 10 unitãøi.

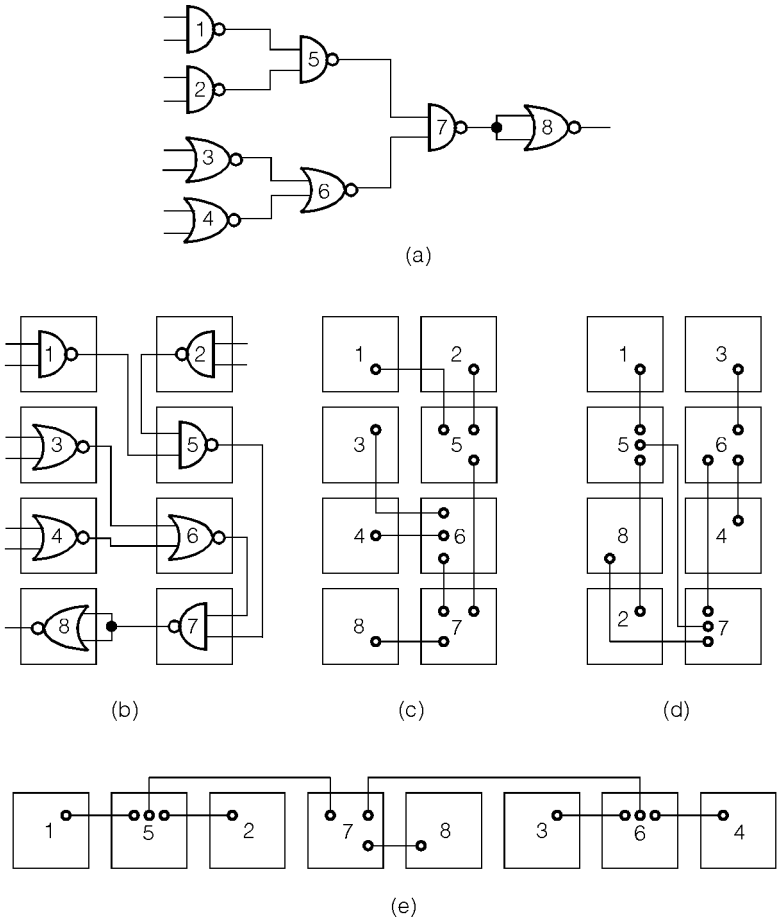

Figura 4.1. (a) Exemplu de circuit. (b) O plasare 2-D a porøilor. (c) O plasare 2-D simbolicã. (d) O plasare 2-D necesitând interconexiuni de 12 unitãøi. (e) O plasare 1-D necesitând interconexiuni de 10 unitãøi.

Lungimea totalã a conexiunilor ω reprezintã o metricã utilizatã pe scarã largã pentru aprecierea calitãøii plasãrii [146]. Considerãm plasarea simbolicã din Figura 4.2(a). Acelaši circuit poate fi plasat ši în modul indicat în Figura 4.2(b), în ultimul caz suprafaøa necesarã fiind însã mai mare. Aceastã suprafaøã constã din douã pãrøi, cea funcøionalã ši cea a conexiunilor. Suma suprafeøelor celulelor funcøionale reprezintã suprafaøa funcøionalã.

Pentru ambele plasãri din Figura 4.2, suprafaøa funcøionalã este aceeaši, fiind diferitã numai suprafaøa conexiunilor. Aceasta din cauza unei distanøe minime care trebuie menøinutã între douã conexiuni ši între o celulã funcøionalã ši o conexiune. De exemplu, considerãm conexiunile (2, 5) ši (8, 9) din Figura 4.2(b). Aceste conexiuni trebuie plasate pe douã piste verticale separate, între care trebuie menøinutã o distanøã minimã.

O plasare care necesitã un spaøiu mare pentru conexiuni va necesita conexiuni de lungime mare, ši deci lungimea totalã a conexiunilor va avea o valoare mare. Deci, lungimea totalã a conexiunilor ω este o metricã potrivitã pentru suprafaøa ocupatã de circuit. Avantajul utilizãrii acestei metrici este cã se poate calcula simplu.

Plasarea modulelor cu scopul de a minimiza lungimea totalã a conexiunilor ω este o problemã NP-completã [146]. Chiar ši cazul cel mai simplu al problemei, ši anume plasarea unidimensionalã, este dificil de soluøionat. Pentru *n* module existã un

număr de $\frac{n!}{2}$ plasări liniare. În practică, numărul modulelor sau al celulelor care trebuie plasate este foarte mare. De aceea, este impracticã enumerarea tuturor configuraţiilor de plasare posibile şi selectarea celei mai bune. Au fost dezvoltate diferite tehnici euristice pentru rezolvarea problemei de plasare. Acestea nu permit de obicei obţinerea soluţiei optime, dar cerinţele de timp ale algoritmilor euristici sunt relativ reduse, reprezentând o funcţie polinomialã de *n*.

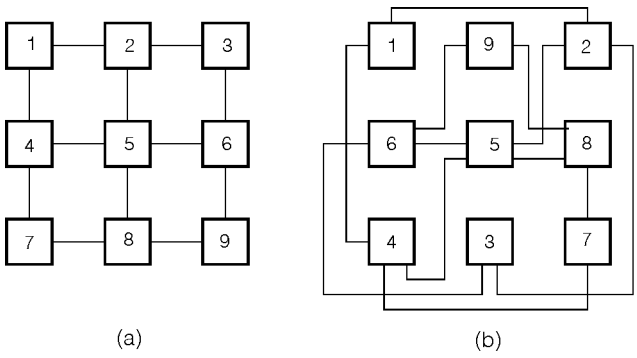

Figura 4.2. (a) Plasare optimã cu $\omega = 12$. (b) Soluţie alternativã cu $\omega = 22$.

4.2 Definirea problemei de plasare

Fiind datã o colecţie de module sau celule care dispun de porturi (pini de intrare, de ieşire, de alimentare şi masã), dimensiunile acestor celule, şi o colecţie de conexiuni (reprezentând seturi de porturi care trebuie conectate), procesul de *plasare* constã din determinarea locaţiil or fizice corespunzãtoare pentru fiecare celulã. Aceste locaţii minimizeazã anumite funcţii obiectiv, cu condiţia respectării unor restricţii impuse de proiectant, de procesul de implementare sau de stilul de proiectare. Exemple de restricţii sunt evitarea suprapunerii celulelor sau cerinţa ca celulele sã fie plasate într-o anumitã suprafaţã rectangularã.

Într-un mod mai formal, problema de plasare poate fi definitã 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 asociazã 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 asociazã un set de module M_{s_i}, unde $M_{s_i} = \{m_j \mid s_i \in S_{m_j}\}$. M_{s_i} se numeşte o conexiune de semnal. Este dat de asemenea şi un set de locaţii $L = \{L_1, L_2, \ldots, L_p\}$, unde $p \geq n$. Problema de plasare este de a asigna fiecare modul $m_i \in M$ unei locaţii unice L_j astfel încât sã fie optimizat un anumit obiectiv.

În mod normal, fiecare modul este considerat zero-dimensional, şi dacã m_i este asignat la locaţia L_j atunci poziţia sa este definitã prin coordonatele (x_j, y_j). Uneori un subset al modulelor din *M* este fixat, deci asignat în prealabil unor locaţii, şi numai modulele rãmase pot fi asignate la locaţiile rãmase neasignate.

4.3 Funcţii de cost şi restricţii

În cadrul procesului de proiectare, etapa de *plasare* este urmatã de cea de *rutare*. O plasare este acceptabilã dacã se poate obţine o rutare de 100% în cadrul suprafeţei date. Funcţia 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*. În secøiunile urmãtoare se prezintã unele tehnici utilizate pentru estimarea lungimii conexiunilor necesare pentru o anumitã plasare.

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. Pentru conexiuni cu doi şi trei pini aceasta este o aproximare exactã. Acestã metodã asigurã o estimare corespunzãtoare pentru cea mai eficientã configuraøie de rutare, care este arborele Steiner. În cazul circuitelor foarte congestionate acestã metodã subestimeazã lungimea conexiunilor.

Metoda aproximãrii arborelui Steiner. Un *arbore Steiner* reprezintã calea cea mai scurtã pentru conectarea unui set de pini. În acestã metodã, pot exista ramificãri din orice punct al unei conexiuni pentru conectarea la aløi pini. Problema determinãrii arborelui Steiner minim este NP-completã. Existã diferite metode pentru determinarea unei aproximãri a arborelui Steiner [3] [45] [52] [84] [91] [99]. Se poate utiliza de asemenea algoritmul lui Lee, prezentat în Capitolul 5.

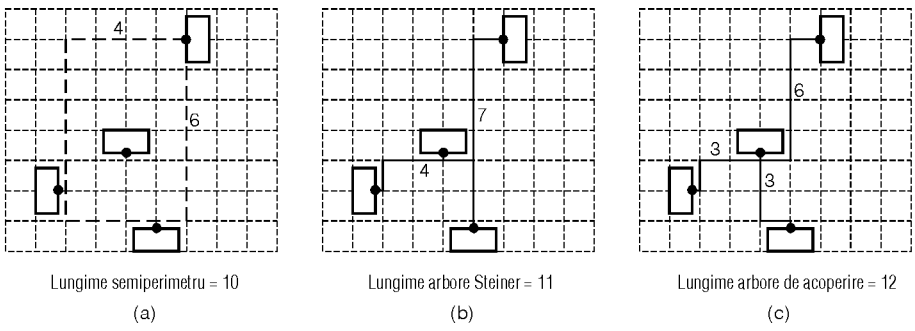

Figura 4.3. Aplicarea diferitelor metode pentru estimarea lungimii conexiunilor.

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 formează cicluri. Un algoritm cu o complexitate polinomială pentru determinarea arborelui de acoperire minim a fost elaborat de Kruskal.

Figura 4.3. prezintă exemple cu diferite metode de interconectare, pentru fiecare indicându-se lungimea conexiunilor.

4.3.2 Minimizarea lungimii totale a conexiunilor

Obiectivul principal al plasării este de a găsi o soluţie care este complet rutabilă. De asemenea, suprafaţa ocupată de conexiunile de rutare trebuie să fie minimă. O metodă pentru a satisface această cerinţă este ca celulele puternic conectate să fie plasate aproape unele de altele. O funcţie obiectiv a cărei minimizare se urmăreşte şi care este des utilizată este $L(P)$, lungimea totală ponderată pentru toate conexiunile de semnal, exprimată ca:

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

unde:

d_n = lungimea estimată a conexiunii n;
w_n = ponderea conexiunii n;
N = setul de conexiuni.

În această estimare, lungimea fiecărei conexiuni este calculată în mod independent de celelalte conexiuni. De aceea, suprafaţa estimată este doar o aproximare grosieră a celei reale.

4.3.3 Minimizarea tăieturii maxime

Considerăm un spaţiu pentru plasare sub forma unei suprafeţe rectangulare (Figura 4.4), 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.

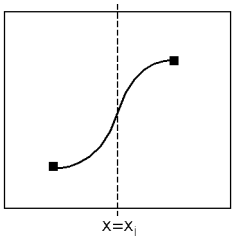

Figura 4.4. O linie de tăietură verticală şi o conexiune intersectată de această linie.

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 tăieturã orizontale y_j și tăietura verticală maximă $Y(P)$ ca fiind:

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

Presupunem cã plasarea se efectuează pentru o rețea de porți, și pentru o plasare datã, $X(P) = 10$ și $Y(P) = 15$. Aceasta înseamnã cã, pentru o anumitã linie de tăieturã verticală v vor fi tãiate 10 conexiuni de această linie. În mod similar, pentru o anumitã linie de tăieturã orizontală h vor fi tãiate 15 conexiuni de această linie. Numărul de canale orizontale de-a lungul liniei de tăieturã v trebuie sã fie de cel puțin 10, deoarece în caz contrar circuitul nu va putea fi rutat utilizând plasarea P. În mod similar, trebuie sã fie cel puțin 15 canale verticale de-a lungul liniei de tăieturã h pentru a se putea realiza rutarea. Existența acestui numãr de canale nu garantează faptul cã rutarea va putea fi executatã complet, fiind deci o condiție necesarã, dar nu și suficientă. Din cele de sus rezultă cã $X(P)$ și $Y(P)$ sunt strâns legate de rutabilitatea rețelei de porți. Dacă se dă o rețea de porți cu H_{max} canale orizontale și V_{max} canale verticale pe o linie de caroiaj, atunci trebuie gãsită o plasare pentru care $X(P) \le H_{max}$ și $Y(P) \le V_{max}$.

Tăieturile $\Phi_P(x_i)$ și $\Phi_P(y_j)$ sunt de asemenea strâns legate de lungimea totală a conexiunilor $L(P)$. Presupunând cã spațierea liniilor de caroiaj este de 1 unitate, se poate arãta cã

$$L(P) = \sum_i \Phi_P(x_i) + \sum_j \Phi_P(y_j) \tag{4.6}$$

unde însumarea se efectuează pentru toate liniile de tăieturã posibile.

Din cele expuse rezultă cã reducerea tăieturii orizontale $X(P)$ și a celei verticale $Y(P)$ prin selectarea unei plasări P corespunzãtoare poate mări probabilitatea rutării rețelei de porți. În plus, minimizarea $X(P)$ și $Y(P)$ poate avea de asemenea o influență benefică asupra lungimii totale a conexiunilor $L(P)$.

În cazul circuitelor care utilizează celule standard sau macro-celule, minimizarea $X(P)$ și $Y(P)$ este de asemenea importantă pentru a se reduce lungimea totală a conexiunilor.

4.3.4 Minimizarea densității maxime

O altă mãsurã a rutabilității unei plasări este *densitatea* $D(P)$, definită după cum urmează. Presupunem cã suprafața disponibilã pentru plasare este divizatã într-o grilă. Figura 4.5(a) ilustrează o rețea de porți cu trei linii și trei coloane. Blocurile funcționale sunt hașurate. În Figura 4.5(b) se aratã separat o porțiune A a suprafeței rezervate pentru conexiuni (un bloc de interconectare). Prin această regiune poate trece un numãr fix de conexiuni orizontale, care se numește capacitate orizontalã a regiunii. În mod similar, se definește și o capacitate verticalã a blocului de interconectare. Figura 4.5(b) prezintã de asemenea un canal B care permite conexiuni verticale. Pentru acest canal este definitã o capacitate verticalã.

Fiind datã o plasare P, este posibilã estimarea numãrului de conexiuni care trebuie sã treacã prin fiecare muchie e_i a unui canal (sau bloc de interconectare). Dacã această estimare este $\eta_P(e_i)$, iar $\psi_P(e_i)$ este capacitatea muchiei e_i, atunci se poate defini densitatea muchiei e_i ca fiind

$$d_P(e_i) = \frac{\eta_P(e_i)}{\psi_P(e_i)} \tag{4.7}$$

Această densitate $d_P(e_i)$ trebuie să fie mai mică decât 1 (sau cel mult 1) în scopul rutabilităţii. Măsura rutabilităţii plasării este dată de:

$$D(P) = \max_i [d_P(e_i)] \tag{4.8}$$

unde maximul se calculează pentru toate muchiile e_i ale regiunilor de rutare (blocuri de interconectare).

4.3.5 Maximizarea performanţelor

Pe măsura avansului tehnologic, viteza de comutare a tranzistoarelor este din ce în ce mai ridicată. Ca urmare a acestei tendinţe, întârzierile datorate interconexiunilor devin importante comparativ cu cele datorate comutării tranzistoarelor. În cazul unor tehnologii ca ECL, acest efect este pronunţat.

Deoarece timpul de comutare al porţilor logice a scăzut la ordinul picosecundelor, frecvenţa ceasului circuitelor VLSI a devenit din ce în ce mai dependentă de propagarea semnalelor prin interconexiunile circuitelor. Pentru numeroase circuite complexe, întârzierile datorate interconexiunilor contribuie cu mai mult de jumătate la ciclul de ceas, şi porţiunea datorată timpului de propagare din cadrul ciclului continuă să crească. Acest fapt are un impact major asupra succesului procesului de proiectare. În etapele de început ale procesului de proiectare nu este posibilă determinarea frecvenţei ceasului numai pe baza caracteristicilor logice şi a întârzierilor de comutare a circuitelor. Timpii mari de propagare, care se datorează caracteristicilor electrice ale interconexiunilor, nu permit obţinerea frecvenţei de ceas aşteptate.

Pentru verificarea şi îmbunătăţirea proprietăţilor temporale ale circuitelor, proiectanţii utilizează utilitare numite analizoare de întârziere sau verificatoare de timp. Acestea permit testarea problemelor care pot apare pe căile scurte sau lungi ale semnalelor.

În timpul procesului de proiectare, întârzierile de propagare ale interconexiunilor nu sunt cunoscute înainte de amplasarea modulelor. Problemele de temporizare pe căile lungi care apar după amplasare sunt foarte dificil de corectat deoarece ele pot necesita nu numai noi iteraţii ale etapelor de proiectare fizică, ci uneori, un număr mare de iteraţii ale etapei de proiectare logică.

Au fost sugerate trei metode generale pentru corectarea problemelor de temporizare pe căile lungi [146]. Prima metodă constă în efectuarea unor modificări în cadrul logicii. De exemplu, întârzierea unei căi poate fi redusă în mod substanţial prin reducerea încărcării unor elemente de circuit de pe această cale.

A doua metodă constă în dimensionarea tranzistoarelor pentru a creşte viteza unor elemente de circuit pe calea lentă. Prin creşterea dimensiunii unora din tranzistoare, întârzierile de comutare, ca şi întârzierile de propagare prin conexiunile pe care se află aceste tranzistoare pot fi reduse substanţial.

O metodă de a creşte viteza unui circuit fără a modifica proiectarea logică a acestuia constă în reducerea timpului de propagare la minim. Acest scop poate fi atins prin impunerea unor constrângeri de temporizare pentru interconexiunile şi căile circuitului. A treia metodă utilizează această tehnică. Au fost raportate diferite încercări pentru ca proiectarea fizică să ţină cont de cerinţele de temporizare. Un circuit VLSI care este optimizat din punct de vedere al temporizării permite o creştere cu 25-35% a frecvenţei ceasului, fără efectuarea unor modificări în proiectarea logică.

Performanţele de viteză ale unui circuit pot fi caracterizate prin întâzierea combinaţională cea mai mare de la un pin de ieşire la un pin de intrare. Dacă întârzierile căilor semnalelor trebuie menţinute sub o anumită valoare maximă, atunci trebuie verificate întârzierile conexiunilor. Deoarece plasarea afectează cerinţele de interconectare ale circuitului, obiectivul unei probleme de plasare poate fi modificat pentru a satisface cerinţele de temporizare ale căilor semnalelor.

4.4 Sinteza metodelor de plasare

Dupã cum s-a arãtat în secøiunea 4.1, problema de plasare este NP-completã. Funcøiile de cost şi restricøiile prezentate în secøiunea precedente nu îmbunãtãøesc această situaøie. Pentru probleme complexe, spaøiul soluøiilor nu permite enumerarea tuturor soluøiilor, motiv pentru care se utilizează tehnici euristice care necesită timpi reduşi de execuøie şi pot gãsi soluøii apropiate de cele optime.

Metodele euristice pentru plasare se pot clasifica în două categorii: *constructive* şi *iterative*.

Metodele constructive realizează plasarea câte unei celule la un moment dat, construind astfel soluøia finală. Aceste metode sunt de două tipuri: metode bazate pe partiøionare şi metode analitice. În ultimul timp, au fost obøinute soluøii de calitate prin combinarea ambelor strategii [67]. Metodele constructive sunt de obicei rapide şi generează soluøii de calitate. Totuşi, acestea sunt în general limitate în privinøa alegerii obiectivelor, şi de multe ori nu obøin optimul global al problemei de plasare.

Metodele iterative au ca scop îmbunãtãøirea soluøiilor existente, în special a celor obøinute prin metode constructive. În general, într-un pas iterativ acestea selectează o subproblemă locală de dimensiuni mici pentru a fi rezolvatã printr-o metodă exactă sau euristică. Aceste metode se împart de asemenea în două categorii, după cum ele aplică tehnici deterministice sau aleatoare.

Metodele iterative deterministice oferă de obicei proiectantului alegerea obiectivelor de plasare, dar de multe ori aceste metode obøin doar un minim local. Timpul necesar obøinerii soluøiei este mult mai redus decât în cazul metodelor aleatoare.

Metodele iterative aleatoare pot accepta, cu o anumită probabilitate redusă, şi soluøii de plasare intermediară de calitate inferioară. Astfel, acestea au posibilitatea de a evita soluøiile optime locale şi de a se apropia de optimul global, dacă dispun de un timp de calcul suficient. Există două metode aleatoare de bază, evoluøia simulată şi călirea simulată.

În literatură a fost publicat un număr mare de algoritmi pentru plasare [37], [57] [67], [68], [71], [82], [87], [88], [104], [109], [113], [119], [124], [139], [141], [151], [152], [153], [189]. Metodele bazate pe partiøionare implică aplicarea recursivă a unui algoritm de partiøionare, de obicei algoritmul Kernighan-Lin sau algoritmul de călire simulată [102]. Pachetul de programe *TimberWolf* [151] pentru plasare şi rutare, care se bazează pe metoda de călire simulată, a fost primul program care a utilizat această metodă pentru problema de plasare. Rezultatele care au fost obøinute sunt foarte bune pentru un număr de câteva mii de celule. Deşi metoda de călire simulată poate obøine soluøia 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 complexitãøii ridicate a algoritmului de călire simulată, au fost propuse mai multe tehnici de creştere a vitezei pentru acest algoritm [107], [119].

Lam şi Delosme [107] au elaborat o schemă îmbunãtãøită de călire, care păstrează rata de acceptare a călirii simulate la o valoare constantă de 44% pe parcursul domeniului intermediar de temperatură. Cu această schemă au fost raportate îmbunãtãøiri atât ale performanøelor, cât şi a timpului de calcul. Această îmbunãtãøire a fost inclusă în versiunile ulterioare ale programului *TimberWolfSC*.

Mallela şi Grover [119] au propus o călire simulată în două etape, utilizând o strategie de grupare de jos în sus. În prima etapă, se aplică o tehnică de grupare. Celulele cu o proximitate mai mare în privinøa numărului de terminale şi a numărului de conexiuni sunt combinate pentru a forma un grup. Ca rezultat al grupării, spaøiul de căutare al călirii simulate este redus în principiu cu dimensiunea grupurilor. Mallela şi Grover raportează o reducere cu un factor de 2 până la 3 a timpului de calcul, şi o reducere cu 6-17% a lungimii conexiunilor, măsurată prin metoda semi-perimetrului.

Această tehnică de călire simulată bazată pe grupare a fost inclusă în sistemul de plasare *LTX2* al Bell Laboratories.

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 propuşi de asemenea pentru soluţionarea problemei speciale de plasare în care toate celulele au aceeaşi dimensiune şi sunt specificate locaţiile posibile ale acestora.

Pentru rezolvarea problemei de plasare liniară, care este o problemă fundamentală pentru proiectarea circuitelor VLSI, au fost propuse diferite metode. Li *et al.* [109] au propus o metodă spectrală în care se utilizează o funcţie obiectiv rezultată dintr-o combinaţie a unei funcţii liniare cu o funcţie cuadratică. Se utilizează astfel atât avantajul unei funcţii obiectiv liniare, care permite obţinerea unei plasări de calitate mai bună în privinţa lungimii conexiunilor, cât şi avantajul unei funcţii cuadratice, care tinde să plaseze componentele mai dispersat, rezultând o soluţie mai fezabilă.

Problema de plasare poate fi transformată într-o problemă de optimizare numerică. Hanan şi Kurtzberg au redus problema de plasare la cea de rezolvare a unui set de ecuaţii liniare pentru a determina locaţiile de echilibru (coordonatele ideale x, y) ale celulelor [146].

Plasarea orientată pe performanţe a fost studiată de numeroşi autori, în special plasarea cu restricţii de timp. Metodele raportate pot fi clasificate în trei categorii [146]. Prima categorie transformă restricţiile de timp de pe căile critice în ponderi ale conexiunilor. Aceste ponderi sunt utilizate pentru a categorisi conexiunile şi a influenţa procedura de plasare. A doua categorie transformă restricţiile de timp ale căilor în limite de timp ale conexiunilor. Aceste limite de timp sunt convertite în limite de lungime ale conexiunilor şi sunt furnizate programului de plasare, care încearcă satisfacerea acestora. A treia metodă constă în furnizarea pentru programul de plasare a unui set de căi critice, împreună cu cerinţele lor de timp. Aceste căi sunt monitorizate în timpul procesului de plasare.

Problema de plasare a circuitelor FPGA a fost abordată prin diferite metode, dintre care şi prin metoda călirii 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 porţiune a spaţiului de pe cip este dedicată 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 bazează pe metoda de călire simulată. Beetem [18] a propus un algoritm de îmbunătăţire iterativă pentru plasarea şi rutarea simultană a circuitelor FPGA. Nag şi Roy [125] a prezentat un algoritm incremental de plasare pentru circuitele FPGA cu o arhitectură bazată pe rânduri de celule, care analizează informaţiile de întârziere a semnalelor şi de rutabilitate pentru a obţine plasări de calitate mai bună. Togawa *et al.* [167] au propus o metodă pentru plasarea şi rutarea simultană a circuitelor FPGA simetrice, metodă bazată pe bipartiţionarea ierarhică.

Gao [82] a elaborat algoritmi de plasare orientaţi pe performanţe bazaţi pe conexiuni şi pe căi de rutare, pentru reţele de porţi, macro-celule şi circuite FPGA Xilinx, cu scopul minimizării întârzierii semnalelor la pinii de ieşire şi a nesimetriei semnalelor la intrările modulelor. În cazul algoritmului bazat pe conexiuni, cerinţele

de întârziere sunt translatate mai întâi în constrângeri de proiectare fizicã, cum sunt constrângeri ale conexiunilor. 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. Unele din problemele apãrute sunt timpii mari de simulare, calitatea redusã a soluøiilor 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. Algoritmul paralel pãstrează calitatea versiunii seriale.

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

Au fost realizate diferite implementãri paralele ale metodei de cãlire simulatã pe diferite tipuri de calculatoare paralele: cu memorie partajatã, cu transmitere de mesaje şi memorie localã, şi calculatoare cu paralelism masiv, ca de exemplu Connection Machine. Kravitz şi Rutenbar [104] au prezentat un algoritm de cãlire simulatã pentru celule standard, care a fost implementat pe un multiprocesor. Aceştia au arãtat cã algoritmul de cãlire simulatã poate fi accelerat în douã moduri pe un multiprocesor cu memorie partajatã: prin executarea mai multor mutãri în paralel, şi prin executarea în paralel a prelucrãrilor necesare pentru fiecare mutare.

Rose *et al.* [141] au conceput euristici pentru plasarea paralelã a celulelor standard cu o calitate echivalentã cu cea a cãlirii simulate. Aceştia au utilizat o metodã rapidã bazatã pe tãietura minimã pentru a evita partea de cãlire lentã la temperaturi înalte. În faza de cãlire la temperaturi joase, spaøiul din cadrul circuitului este partiøionat şi este asignat diferitelor procesoare, astfel încât fiecare procesor mută celule într-o anumitã zonã şi, ori de câte ori o mutare este acceptatã, transmite rezultatul tuturor procesoarelor.

În continuare se descriu într-un mod mai detaliat unele metode de plasare. În secøiunea 4.4.1 se prezintã plasarea prin metode constructive. În secøiunea 4.4.2 se descrie un algoritm de plasare care utilizează în mod repetat metoda de partiøionare pe baza tãieturii minime. În secøiunea 4.4.3 se descrie modul de utilizare a algoritmului de cãlire simulatã pentru plasare, şi se prezintã algoritmul de plasare utilizat de pachetul de programe TimberWolf. În secøiunea 4.4.4 se prezintã plasarea prin partiøionarea ierarhicã, în care se utilizează metoda de partiøionare pe baza tãieturii minime. În secøiunea 4.4.5 se prezintã plasarea prin metode de optimizare numericã, fiind descrisã o metodã numitã plasare controlatã de forøe. În secøiunea 4.4.6 se prezintã utilizarea metodelor spectrale pentru rezolvarea problemei de plasare liniarã. În secøiunea 4.4.7 se descriu pe scurt alte metode iterative de plasare: asignarea cuadraticã, optimizarea reøelelor rezistive, algoritmul Steinberg, metoda spaøiului grafurilor, şi plasarea bazatã pe operaøii cu linii şi coloane.

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 consideră ca exemplu următorul algoritm pentru plasarea a *n* module într-o linie. Suprafaţa pe care se amplasează modulele este divizată în *n* locaţii, iniţial fiecare locaţie fiind liberă. În fiecare iteraţie a algoritmului, se va plasa un modul într-una din locaţiile libere. La sfârşitul unei iteraţii, va exista o plasare parţială a unui subset de module. Există două decizii care trebuie luate în fiecare iteraţie. Acestea se referă la selectarea modulului care va fi adăugat la plasarea parţială şi la locaţia în care se va plasa modulul selectat.

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*.

Modulul selectat poate fi plasat în oricare din cele (*n* - *i*) locaţii libere. Se poate estima modificarea funcţiei de cost pentru fiecare din cele (*n* - *i*) opţiuni, alegând opţiunea care este cea mai avantajoasă. De exemplu, dacă funcţia de cost este lungimea totală a conexiunilor, este selectată locaţia care va determina creşterea minimă a lungimii conexiunilor.

Pentru a obţine o plasare parţială iniţială, se poate selecta un singur modul, în mod aleator sau pe baza unui criteriu euristic. Acest modul poate fi plasat, de exemplu, în locaţia din mijloc. Este logic ca iniţial să fie selectat modulul cu conectivitatea maximă.

În continuare se vor descrie într-un mod mai formal cele două etape ale unui algoritm de plasare constructivă.

Pentru selecţia unui modul neplasat, fie *X* setul modulelor plasate, set în care unele module sunt fixe (module hard), ca de exemplu celule de I/E, şi fie *Y* setul modulelor neplasate. În mod intuitiv, următorul modul candidat din setul *Y* trebuie să fie cel cu cele mai multe conexiuni cu modulele din setul *X*, pentru a se obţine lungimea de rutare cea mai mică. Regulile de selecţie a modulului candidat vor fi generate pe baza matricii de capacitate **C**, după cum urmează:

$$M_Y = \max_{y \in Y} \left\{ \sum_{x \in X} c_{yx} \right\} \tag{4.10}$$

unde *x* este un modul din setul *X* al modulelor plasate, iar *y* este un modul care trebuie plasat, aparţinând setului *Y* al modulelor neplasate. În cazul unei indecizii, deci atunci când poate fi ales mai mult decât un candidat, se poate selecta în mod arbitrar oricare din acei candidaţi.

De exemplu, considerãm urmãtoarea matrice de capacitate **C** pentru modulele *A*, *B*, *C*, *D* øi *E*:

$$\mathbf{C} = \begin{bmatrix} 0 & 4 & 2 & 3 & 5 \\ 4 & 0 & 0 & 5 & 4 \\ 2 & 0 & 0 & 0 & 1 \\ 3 & 5 & 0 & 0 & 2 \\ 5 & 4 & 1 & 2 & 0 \end{bmatrix} \tag{4.11}$$

Dacã modulele *A*, *B* øi *E* au fost plasate, urmãtorul modul care trebuie plasat este *D*, deoarece $M_D = 3 + 5 + 2 = 10 > M_C = 2 + 1 = 3$.

Mai general, dacã se considerã cã ponderea este un factor important al setului de semnale, se poate introduce o pondere w_{ij} pentru conexiunea dintre modulele *i* øi *j*. O formã generalã a regulii de selecøie este urmãtoarea:

$$M_Y = \max_{y \in Y} \left\{ \sum_{x \in X} q_{yx} \right\} \tag{4.12}$$

unde **Q** = $[q_{ij}]$, $q_{ij} = c_{ij}\, w_{ij}$. Dacã $w_{ij} = 1$, nu existã deosebire între matricile **C** øi **Q**.

Existã diferite alte reguli de selecøie [189]. De exemplu, o altã regulã simplã de selecøie poate fi bazatã pe matricea de capacitate, alegând ca øi candidat modulul cu

$$\max_i \left\{ \sum_j c_{ij} \right\}, \ \forall\, j \in V \tag{4.13}$$

unde modulul *i* aparøine setului de module neplasate, iar *V* este setul de noduri ale grafului *G* al circuitului.

Pentru plasarea modulului selectat, trebuie gãsite coordonatele unui punct optim pentru care lungimea totalã de rutare este minimã. Dupã cum s-a indicat în secøiunea 4.1, lungimea de rutare poate fi mãsuratã prin distanøa euclidianã sau prin distanøa Manhattan. Dacã existã *m* module fixe M_j aflate în poziøiile (x_j, y_j), unde x_j øi y_j sunt întregi, $j = 1, 2, \ldots, m$, øi un modul *M* în poziøia (x, y) care poate fi mutat, cu o capacitate c_{Mj} între *M* øi modulul fix M_j, $j = 1, 2, \ldots, m$, lungimea totalã de rutare a modulului *M* la toate modulele fixate poate fi exprimatã în metrica euclidianã astfel:

$$D = \sum_{j=1}^{m} c_{Mj} [(x - x_j)^2 + (y - y_j)^2] \tag{4.14}$$

iar în metrica Manhattan astfel:

$$D = \sum_{j=1}^{m} c_{Mj} [|x - x_j| \ + \ |y - y_j|] \tag{4.15}$$

Pentru a gãsi punctul optim (x_{opt}, y_{opt}) pentru care lungimea totalã de rutare *D* este minimã, se vor prezenta separat cazurile celor douã metrici.

A. Metrica euclidianã. Se observã cã în ecuaøia (4.14), cele douã componente din membrul drept sunt independente. Deci, lungimea totalã în direcøia *x* este independentã de lungimea totalã în direcøia *y*, care este cea mai importantã pentru ordonarea liniarã sau plasare. Astfel, cele douã componente pot fi considerate separat. Se considerã lungimea totalã în direcøia *x*:

$$D_x = \sum_{j=1}^{m} c_{Mj} (x - x_j)^2 \tag{4.16}$$

Dacă se calculează derivata lui D_x faţă de x şi se egalează cu zero, se obţine:

$$x_{opt} = \frac{\sum_{j=1}^{m} c_{Mj} x_j}{\sum_{j=1}^{m} c_{Mj}} \tag{4.17}$$

care este punctul optim pentru obţinerea valorii minime pentru D_x. Similar,

$$y_{opt} = \frac{\sum_{j=1}^{m} c_{Mj} y_j}{\sum_{j=1}^{m} c_{Mj}} \tag{4.18}$$

Se defineşte:

$$x_{mopt} = [x_{opt} + 1] \tag{4.19}$$

$$y_{mopt} = [y_{opt}] \tag{4.20}$$

dacă modulul M_m este în legătură numai cu unul din modulele plasate, iar în caz contrar:

$$x_{mopt} = [x_{opt} + 0.5] \tag{4.21}$$

$$y_{mopt} = [y_{opt} + 0.5] \tag{4.22}$$

unde (x_{mopt}, y_{mopt}) este numit *punct optim local* pentru modulul M_m, iar [x] este cel mai mare întreg mai mic sau egal cu x.

B. Metrica Manhattan. Înainte de a determina punctul optim pentru lungimea minimă de rutare definită în ecuaţia (4.15), se consideră următoarea problemă generalizată: Fiind date n puncte i (i = 1, 2, …, n) pe o linie, şi un punct mobil x cu capacitatea c_{ix} între i şi x, să se determine poziţia $x = x_{opt}$ astfel încât

$$\sum_{i=1}^{n} c_{ix} |x - i| \tag{4.23}$$

să fie minim. Această problemă este un caz special al problemei de găsire a unui centru absolut pentru un graf general. O condiţie suficientă pentru determinarea punctului optim xopt este dată de următoarea teoremă.

Teorema 4.4.1.1. Punctul x_{opt} este optim dacă este satisfăcută inegalitatea:

$$\sum_{i=1}^{x_{opt}-1} c_{ix} (x_{opt} - i) \le \frac{N}{2} \le \sum_{i=1}^{x_{opt}} c_{ix} (x_{opt} - i) \tag{4.24a}$$

unde:

$$N = \sum_{i=1}^{n} c_{ix} \tag{4.24b}$$

Revenind la ecuaţia (4.15), se consideră că partea dreaptă a acesteia este formată din doi termeni independenţi. Lungimea totală de rutare pentru direcţia x este:

$$D_x = \sum_{j=1}^{m} c_{Mj} |x - x_j| \tag{4.25}$$

Fie $x_{max} = \max_j \{x_j\}$, $x_{min} = \min_j \{x_j\}$, $j = 1, 2, \ldots, m$. De notat cã $x = x_{opt}$ trebuie sã fie în domeniul:

$$x_{min} \le x_{opt} \le x_{max} \tag{4.26}$$

Se defineşte:

$$x(k) = x_j \tag{4.27}$$

$$c(k) = c_{Mj} \tag{4.28}$$

unde $x(k)$ şi $c(k)$ sunt formele sortate ale lui x_j, respectiv c_{Mj}, iar $x(k)$ este în ordine ascendentã, $j = 1, 2, \ldots, m$; $k = x_{min}, \ldots, x_{max}$. Relaøia dintre j şi k este determinatã de procedura de sortare.

Din Teorema 4.4.1.1, o condiøie suficientã pentru determinarea punctului optim x_{opt} este:

$$\sum_{k=x_{\min}}^{x_{opt}-1} c(k)[x_{opt} - x(k)] \le \frac{N}{2} \le \sum_{k=x_{\min}}^{x_{opt}} c(k)[x_{opt} - x(k)] \tag{4.29a}$$

unde:

$$N = \sum_{x_{\min}}^{x_{\max}} c(k) \tag{4.29b}$$

Similar, y_{opt} este dat de:

$$\sum_{k=x_{\min}}^{y_{opt}-1} c(k)[y_{opt} - y(k)] \le \frac{N}{2} \le \sum_{k=x_{\min}}^{y_{opt}} c(k)[y_{opt} - y(k)] \tag{4.30a}$$

unde:

$$N = \sum_{y_{\min}}^{y_{\max}} c(k) \tag{4.30b}$$

iar $y(k)$ este forma sortatã a lui y_j în ordine ascendentã, cu:

$$y(k) = y_j \tag{4.31}$$

Relaøia dintre j şi k este determinatã de procedura de sortare pentru $j = 1, 2, \ldots, m$ şi $k = x_{min}, \ldots, x_{max}$.

Dacã $x_{mopt} = x_{opt}$, $y_{mopt} = y_{opt}$, poziøia (x_{mopt}, y_{mopt}) este numitã *punct optim local* pentru modulul M_m în metrica Manhattan.

Pânã acum s-a indicat cum este selectat urmãtorul modul candidat dacã existã deja module plasate, şi felul în care este plasat acest modul. Problema care se pune este selectarea şi plasarea primului modul.

Dacã existã module hard, ca module de I/E sau alte module specificate care sunt fixate în anumite poziøii, acestea vor fi plasate primele. Dacã nu existã module hard, existã mai multe metode pentru selectarea primului modul. Se pot lua în considerare urmãtoarele criterii:

1. Este selectat modulul cu gradul maxim, care este plasat în centru sau într-o altã poziøie adecvatã.
2. Este selectat modulul cu gradul minim, care este plasat într-un colø sau într-o altã poziøie adecvatã.

Urmãtoarea problemã este plasarea modulului a cãrei poziøie optimã localã este ocupatã de un alt modul fixat. În principiu, este posibilã încercarea tuturor pozi-

ţiilor disponibile şi alegerea poziţiei celei mai bune. Acest proces este însă consumator de timp, şi nu este avantajoasă utilizarea lui pentru plasarea iniţială. Este necesară însă testarea poziţiilor din jurul poziţiei optime locale. Chiar dacă poziţia optimă a modulului candidat nu se află în jurul poziţiei optime locale, această metodă este mult mai rapidă şi mai eficientă din cauza faptului că matricea de capacitate este rară.

Se prezintă în continuare un algoritm pentru plasarea constructivă iniţială, elaborat de Wang şi Chen [189].

1. Pe baza unui set dat S de seturi de semnale, se conectează fiecare set de semnale printr-un arbore.
2. Se determină matricea de capacitate.
3. Se testează dacă există module hard. În caz afirmativ, se plasează modulele hard în poziţiile cerute, apoi se continuă cu pasul 5. În caz contrar, se continuă cu pasul 4.
4. Se alege primul modul M_0 şi se plasează în poziţia selectată.
5. Se selectează un modul candidat.
6. Se determină punctul optim local W_{mopt} al modulului candidat.
7. Se testează dacă W_{mopt} este ocupat. Dacă da, se testează setul care conţine W_{mopt} şi se plasează modulul în poziţia optimă. În caz contrar, se plasează modulul în punctul optim local.
8. Se testează dacă M_m este ultimul modul. Dacă da, algoritmul se termină. În caz contrar, se continuă cu pasul 5.

Există diferite alte metode constructive. În secţiunea 4.4.5 va fi descrisă o metodă numerică numită plasare bazată pe forţe.

4.4.2 Plasarea pe baza tăieturii minime

În capitolul 3 a fost prezentată problema de partiţionare a circuitelor. Un algoritm de partiţionare încearcă gruparea modulelor puternic conectate. O asemenea grupare va reduce de asemenea lungimea interconexiunilor. Se va descrie în continuare un algoritm care utilizează în mod repetat o procedură de partiţionare prin metoda tăieturii minime pentru generarea unei plasări.

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 tăieturâ verticală c_1, şi se plasează modulele din blocul B la dreapta liniei de tăieturâ c_1. Setul de tăieturâ 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 tăieturâ verticală c_2. Similar, blocul B se partiţionează în două blocuri B_1 şi B_2, utilizând o linie de tăieturâ verticală c_3 (Figura 4.5).

Acest proces poate fi repetat prin introducerea altor linii de tăieturâ. Presupunem că procedura de partiţionare utilizată generează o partiţie optimă. Pentru întregul circuit, $\Phi_P(c_1)$ este tăietura minimă posibilă. Similar, $\Phi_P(c_2)$ este tăietura minimă

posibilã pentru subcircuitul *A*. Nu se poate scrie cã $\Phi_P(c_{i+1}) \le \Phi_P(c_i)$, $1 \le i \le r$ - 1, deši în acest caz s-ar putea obøine minimizarea funcøiei $X(P)$.

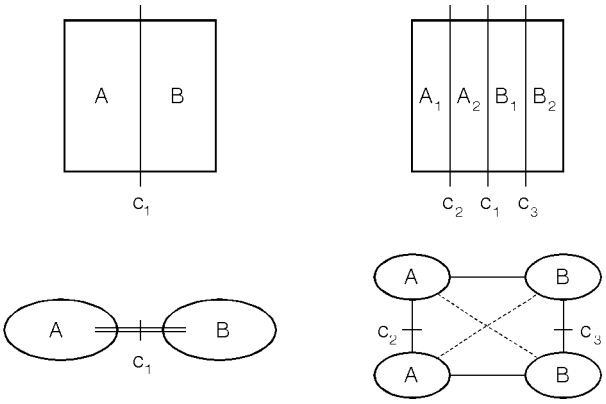

Figura 4.5. Utilizarea partiøionãrii pentru a reduce $X(P)$.

Procedura descrisã mai sus nu minimizeazã $X(P)$, ci minimizeazã $\Phi_P(c_2)$ cu condiøia ca $\Phi_P(c_1)$ sã fie minim. Aceastã funcøie se poate scrie ca $\Phi_P(c_2) \mid \Phi_P(c_1)$. Procedura minimizeazã de asemenea $\Phi_P(c_3) \mid \Phi_P(c_1)$.

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

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

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. Cerinøele principale ale algoritmului sunt:

1. O procedurã eficientã pentru partiøionarea circuitului, ši
2. O strategie de selecøie a liniilor de tãieturã.

Funcøia $F(P)$ (ecuaøia 4.32) este minimizatã prin partiøionarea circuitului mai întâi în douã, astfel încât numãrul de conexiuni care traversează c_1 este minimizat. Dacã c_1 este o linie de tãieturã verticalã, atunci celulele din stânga liniei c_1 sunt fixate ši nu se pot muta în dreapta. Celulele din dreapta liniei c_1 sunt de asemenea restricøionate ši nu se pot muta în stânga. Apoi, se utilizează următoarea linie de tãieturã c_2, ši conexiunile care traversează c_2 sunt minimizate øinând cont de restricøia impusã deja de c_1. Procedura se continuã până când vor fi utilizate toate cele r linii de tãieturã. Din cauza faptului cã aceastã procedurã este de tip *greedy*, soluøia obøinutã nu este în mod garantat cea optimã global.

Existã trei metode mai utilizate pentru selecøia liniilor de tãieturã ši a ordinii în care ele sunt prelucrate [146]. În cazul primei metode, numitã *procedura de plasare cuadraticã*, suprafaøa de plasare este divizatã în patru unitãøi prin douã linii de tãieturã, una verticalã ši cealaltã orizontalã, ambele trecând prin centru. Procedura de divizare anterioarã este aplicatã apoi recursiv fiecãrui sfert din suprafaøã, până când întreaga suprafaøã este divizatã în locaøii. Această secvenøã este ilustratã în Figura 4.6(a), fiind avantajoasã pentru circuite cu o densitate mare de rutare în centru.

În a doua metodă, numită *procedura de plasare prin bisecţie*, suprafaţa este divizată în mod repetat în jumătăţi egale prin linii de tăietură orizontale, obţinându-se segmente orizontale. Această procedură de divizare este continuată până când fiecare segment orizontal este o linie, celulele fiind asignate acestor linii. Apoi, fiecare linie este bisecţionată vertical în mod repetat, până când subregiunile rezultate conţin o locaţie. Această metodă este avantajoasă pentru plasarea celulelor standard şi este ilustrată în Figura 4.6(b).

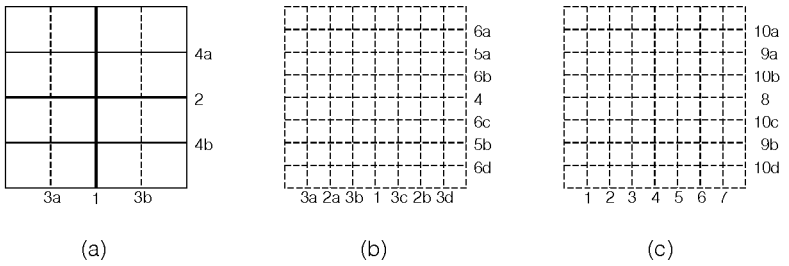

Figura 4.6. Ilustrarea secvenţelor de aplicare a liniilor de tăietură.

În cazul ultimei metode, cele *n* celule ale circuitului sunt divizate prin linia de tăietură c_1 în două seturi de *k* şi (*n* - *k*) celule, astfel încât $\Phi_P(c_1)$ să fie minimizat. Primele *k* celule obţinute sunt asignate liniei de sus (sau celei de jos). Procedura este aplicată apoi pentru cele (*n* - *k*) celule, care sunt divizate în *k* şi (*n* - 2*k*) celule. Procesul este continuat până când toate celulele sunt asignate liniilor. Celulele sunt asignate apoi coloanelor utilizând bisecţia verticală. Această secvenţă este ilustrată în Figura 4.6(c), fiind recomandată pentru celulele cu un număr mare de interconexiuni la periferie.

Structura unui algoritm recursiv de plasare pe baza tăieturii minime este prezentată în Figura 4.7.

```
Algorithm Tăietură_min (S, n, C);
        /* S este suprafaţa de plasare                    */
        /* n este numărul celulelor care trebuie plasate  */
        /* n0 este numărul celulelor dintr-o locaţie      */
        /* C este matricea de conectivitate               */
begin
        if (n ≤ n0) then plasare_celule (S, n, C)
        else
                (S1, S2) ← diviz_supr (S);
                (n1, c1), (n2, c2) ← partit (n, C);
                Tăietură_min (S1, n1, c1);
                Tăietură_min (S2, n2, c2);
        endif
end.
```

Figura 4.7. Structura unui algoritm de plasare recursiv pe baza tăieturii minime.

Procedura de partiţionare prezentată anterior nu ţine cont de poziţia pinilor de I/E (care este de obicei fixă) şi de semnalele care intră într-un grup de celule. Aceste semnale afectează poziţia în care trebuie plasate celulele în aceeaşi măsură în care afectează această poziţie conexiunile din cadrul grupului. Includerea acestor semnale în procedura de plasare bazată pe partiţionare se numeşte *propagarea terminalelor* [68].

4.4.3 Plasarea prin metoda călirii simulate

Metoda călirii simulate este una din cele mai utilizate pentru plasarea celulelor. Algoritmul general a fost prezentat în capitolul 3, secøiunea 3.4.4. În aceastã secøiune se prezintã modul în care algoritmul poate fi adaptat pentru plasare. Se prezintã de asemenea algoritmul utilizat de pachetul de programe pentru plasare şi rutare TimberWolf, dezvoltat de Carl Sechen şi Sangiovanni-Vincentelli [151]. Acesta a fost primul program care a aplicat metoda de călire simulatã pentru problema de plasare.

4.4.3.1 Aplicarea algoritmului de călire simulată pentru plasare

Algoritmul de călire simulatã poate fi modificat pentru plasarea celulelor prin alegerea unei funcøii adecvate de *perturbaøie* pentru a genera o nouã configuraøie de plasare, şi prin definirea unei funcøii corespunzãtoare de *acceptare*. Pentru simplitate, se considerã cã suprafaøa de plasare este modelatã sub forma unei table de şah, şi fiecare celulã a circuitului poate fi plasatã într-un pãtrat al tablei. O funcøie simplã de vecinãtate se obøine prin interschimbarea perechilor, în care sunt alese douã locaøii şi 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 şi 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 şi a parametrilor furnizaøi de utilizator, programul de plasare construieşte o topologie de circuit cu celule standard. Acești 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 dupã cum este necesar, lungimea conexiunilor este minimizatã din nou, şi se executã rutarea globalã preliminarã. În a treia etapã, sunt efectuate modificãri locale ale plasãrii pentru a reduce numãrul pistelor de rutare necesare. Se va prezenta în continuare prima etapã a algoritmului, care utilizează călirea simulatã pentru plasare.

Funcøia obiectiv care este minimizatã de algoritmul TimberWolf3.2 în timpul plasãrii este costul estimat al interconexiunilor. Scopul primei etape este gãsirea unei plasãri a celulelor standard astfel încât costul total estimat al interconexiunilor sã fie minimizat. Se utilizează o funcøie de vecinãtate numitã *generate* pentru a produce noi stãri prin efectuarea unei selecøii aleatoare din trei *funcøii de perturbaøie* posibile:

1. Mutarea unei singure celule într-o nouã locaøie, de exemplu într-un rând diferit.
2. Interschimbarea a douã celule.
3. Oglindirea unei celule dupã axa *x*.

Perturbaţiile sunt limitate la o regiune din cadrul unei ferestre de înălţime H_T şi lăţime W_T. De exemplu, dacă o celulă trebuie mutată, locaţia destinaţie este aleasă în cadrul unei ferestre de limitare centrată în jurul celulei (Figura 4.8).

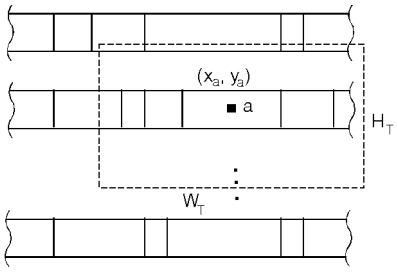

Figura 4.8. Fereastră de limitare centrată în jurul unei celule.

Două celule *a* şi *b*, centrate în poziţiile (x_a, y_a) şi (x_b, y_b) vor fi selectate pentru interschimbare numai dacă $|x_a - x_b| \le W_T$ şi $|y_a - y_b| \le H_T$. Dimensiunile ferestrei sunt funcţii descrescătoare cu temperatura *T*. Dacă temperatura curentă este T_1 şi următoarea temperatură este T_2, lăţimea şi înălţimea ferestrei sunt micşorate astfel:

$$W(T_2) = W(T_1)\frac{\log(T_2)}{\log(T_1)} \tag{4.33}$$

$$H(T_2) = H(T_1)\frac{\log(T_2)}{\log(T_1)} \tag{4.34}$$

Funcţia de cost utilizată de algoritmul TimberWolf3.2 este suma a trei componente:

$$\gamma = \gamma_1 + \gamma_2 + \gamma_3 \tag{4.35}$$

Componenta γ_1 este o măsură a lungimii totale de interconectare estimate. Pentru fiecare conexiune *i*, dacă întinderea pe orizontală şi pe verticală este X_i, respectiv Y_i, atunci lungimea estimată a conexiunii *i* este (X_i + Y_i). Această valoare trebuie multiplicată cu ponderea w_i a conexiunii. Unei conexiuni i se pot ataşa două ponderi, o componentă orizontală w_i^H şi o componentă verticală w_i^V . Deci,

$$\gamma_1 = \sum_{i \in N} [w_i^H \cdot X_i + w_i^V \cdot Y_i] \tag{4.36}$$

unde însumarea se efectuează pentru toate conexiunile *i* din setul de conexiuni *N*.

Ponderea unei conexiuni este utilă pentru a indica măsura în care o conexiune este critică. Dacă o conexiune face parte dintr-o cale critică, de exemplu, trebuie ca aceasta să fie cât mai scurtă pentru a introduce o întârziere cât mai mică. Se pot mări ponderile conexiunilor critice pentru a se atinge acest scop. Ponderile independente pe orizontală şi pe verticală permit utilizatorului flexibilitatea de a favoriza conexiunile într-o direcţie faţă de cealaltă direcţie.

Atunci când o celulă este mutată sau două celule sunt interschimbate, este posibil să existe o suprapunere între două sau mai multe celule. Fie O_{ij} suprafaţa de suprapunere între două celule *i* şi *j*. Suprapunerile nu sunt de dorit şi trebuie minimizate. A doua componentă a funcţiei de cost, γ_2, este interpretată ca penalizare a suprapunerilor, şi este definită astfel:

$$\gamma_2 = w_2 \sum_{i \ne j} [O_{ij}]^2 \tag{4.37}$$

În ecuaøia de sus, w_2 este ponderea penalizării. Prin ridicarea la pătrat a suprapunerii se asigurã penalizări foarte mari pentru suprapuneri mai mari.

Datoritã mutării celulelor şi a interschimbării acestora, lungimea unui rând de celule poate deveni mai mare sau mai micã. A treia componentã a funcøiei de cost reprezintã o penalizare pentru cazul în care lungimea unui rând R depãşeşte lungimea aşteptatã $\overline{L_R}$ sau este mai scurtã decât aceasta:

$$\gamma_3 = w_3 \sum_{rânduri} |L_R - \overline{L_R}| \tag{4.38}$$

unde w_3 este ponderea diferenøei de lungime. Distribuøia inegalã a lungimii rândurilor are ca efect irosirea spaøiului. Existã de asemenea o legãturã între lungimea totalã a conexiunilor şi rutabilitate, pe de o parte, şi distribuøia inegalã, pe de altã parte.

Funcøia de rãcire este reprezentatã de

$$T_{i+1} = \alpha(T_i) \times T_i \tag{4.39}$$

unde $\alpha(T)$ este parametrul ratei de rãcire, determinat experimental. Procesul de călire începe la o temperaturã iniøialã foarte ridicatã, de exemplu 4×10^6. Iniøial, temperatura este redusã rapid [$\alpha(T) \approx 0.8$]. În domeniul temperaturilor medii, temperatura este redusã lent [$\alpha(T) \approx 0.95$].]. În domeniul temperaturilor joase, temperatura este redusã din nou rapid [$\alpha(T) \approx 0.8$].. Algoritmul se terminã atunci când $T < 1$.

La fiecare temperaturã, se încearcã efectuarea unui numãr fix de mutări. Numărul optim al mutărilor depinde de dimensiunea circuitului. De exemplu, pentru un circuit cu 200 de celule, se recomandã 100 de mutări pe celulã, ceea ce presupune evaluarea a 2.34×10^6 configuraøii în aproximativ 125 de trepte de temperaturã. Pentru un circuit cu 3000 de celule, se recomandã 700 de mutări pe celulã, ceea ce conduce la un numãr total de 247.5×10^6 încercări [146].

4.4.4 Plasarea prin partiøionare ierarhicã

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. Fiecare nod intern reprezintã un subcircuit partiøionat la un anumit nivel. 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 integreazã 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.

Algoritmul de plasare prin partiøionare ierarhicã este prezentat în Figura 4.9.

```
Algorithm Plasare_ierarhică;
        /* Partiţionare ierarhică şi plasare   */
begin
        Se calculează numărul de nivele necesare pentru partiţionare nr_nivele;
        nivel = 0;
        for (nivel < nr_nivele - 1) do
                for (toate ferestrele de la acest nivel) do
                        sel_fereastră (nivel, index_fereastră);
                        Partiţionare multinivel prin tăietura proporţională;
                        TimberWolfSC;
                        ajustare_rânduri (ρ);
                        nivel ++;
                endfor
        endfor
        /* Plasare finală              */
        for (toate ferestrele de la nivelul inferior) do
                Călire simulată la temperatură joasă prin TimberWolfSC;
        endfor
end.
```

Figura 4.9. Algoritmul de plasare prin partiţionare ierarhică.

Prima fază a algoritmului este compusă din patru paşi principali. În primul pas este selectată o fereastră din cadrul circuitului, pentru care configuraţia de plasare este rafinată în mai multe iteraţii. În al doilea pas, celulelor selectate li se aplică partiţionarea ierarhică pe baza tăieturii proporţionale, care generează grupuri în cadrul circuitului. În al treilea pas, grupurile sunt transformate în super-celule, pentru care se aplică programul de plasare TimberWolfSC. În ultimul pas, se efectuează ajustarea rândurilor pe care sunt plasate celulele, pentru a elimina distribuţia neuniformă a grupurilor. În următoarea iteraţie a algoritmului, fereastra este deplasată în următoarea poziţie, care se suprapune peste ferestrele procesate anterior de pe acelaşi nivel.

În ultima fază, cea de rafinare, grupurile sunt transformate în celule individuale, şi sunt plasate utilizând călirea simulată la o temperatură joasă.

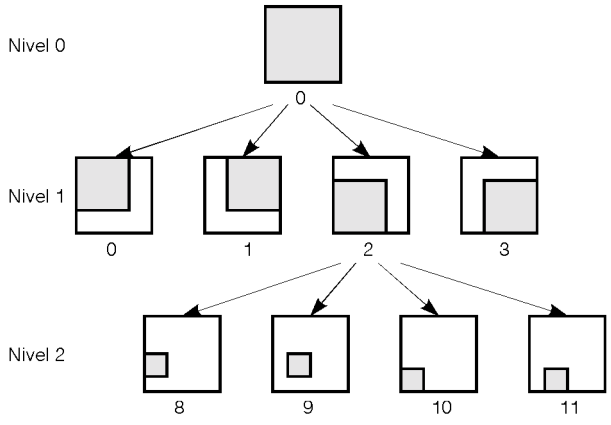

Figura 4.10. Plasare ierarhică prin metoda ferestrelor mobile.

Prima fereastră acoperă întregul circuit, şi dimensiunea ferestrei se reduce pe măsură ce se ajumge la nivelele inferioare din ierarhie. Numărul de grupuri acoperite de o fereastră este menţinut sub o anumită valoare constantă, numită capacitatea ferestrei. Poziţia unui grup reprezintă poziţia medie a celulelor din cadrul grupului.

În fiecare nivel, ferestrele sunt baleiate de la coløul din stânga sus al circuitului spre coløul din dreapta jos, de la un rând orizontal la altul. Figura 4.10 ilustrează utilizarea ferestrei în procesul de plasare. La nivelul 0, fereastra acoperã întregul circuit. La nivelul 1, fiecare din cele patru ferestre acoperã 43% din circuit, şi ele se suprapun reciproc. La nivelul 2, o fereastrã a nivelului 1 este acoperitã de patru ferestre ale nivelului 2, deci existã 16 ferestre la nivelul 2.

Acest concept al ferestrelor, combinat adesea cu o tehnicã de partiøionare succesivã, a fost utilizat de numeroase programe de plasare. De exemplu, programul PROUD, elaborat de Tsay şi Kuh, utilizează bisecøia recursivã a circuitului în fiecare etapã de partiøionare. Ceea ce este caracteristic pentru metoda utilizatã de Hamada *et al.* [88] este faptul cã circuitul este acoperit de o secvenøã de ferestre, în care douã ferestre consecutive sunt suprapuse. Suprapunerea între ferestre permite trecerea celulelor peste limitele partiøiilor, ceea ce poate compensa efectul partiøiei precedente în cazul în care aceasta a afectat calitatea plasării.

Ajustarea rândurilor cuprinde ca operaøie principalã modificarea lungimii acestora prin mutarea celulelor dintr-un rând mai lung în rânduri adiacente mai scurte. Algoritmul procedurii de ajustare este prezentat în Figura 4.11.

```
Procedure ajustare_rânduri (ρ);
begin
        while (1) do
                Se selectează rândul cel mai lung rând_lung şi un rând
                        adiacent mai scurt rând_scurt;
                Se alege o celulă C din rând_lung şi se calculează costul combinat:
                        δ = ρ · Δlungime_rând + Δlungime_conexiune;
                if (δ > 0) then exit;
                else
                        Se mută celula C din rând_lung în rând_scurt;
                endif
        endwhile
end
```

Figura 4.11. Procedura pentru ajustarea lungimii rândurilor.

Interschimbarea celulelor trebuie efectuatã cu precauøie, deoarece aceasta egalizează lungimea rândurilor în detrimentul lungimii totale a conexiunilor. Egalizarea lungimii rândurilor se terminã atunci când minimul costului combinat

$$\delta = \rho \cdot \Delta lungime\_rând + \Delta lungime\_conexiune; \qquad (4.40)$$

devine pozitiv. Primul termen al ecuaøiei este negativ, deoarece trebuie minimizatã diferenøa dintre lungimile rândurilor. Al doilea termen, care este lungimea conexiunilor sacrificatã de egalizarea lungimii rândurilor, este pozitiv. Factorul ρ ponderează $\Delta lungime\_rând$. Dacã ρ este setat la o valoare mare, lungimea rândurilor poate fi egalizatã într-o mare mãsurã, cu preøul creşterii lungimii conexiunilor. În caz contrar, egalizarea lungimii rândurilor sacrificã doar o micã porøiune a lungimii conexiunilor.

În [88] s-a raportat pentru algoritmul de plasare prin partiøionare ierarhicã o reducere medie a lungimii totale a conexiunilor de 2.49% faøã de programul TimberWolfSC, versiunea 5.6, şi o reducere medie a timpului de calcul de 18.3% faøã de acelaşi program. S-a constatat de asemenea cã timpul de calcul creşte liniar cu dimensiunea circuitului [88].

4.4.5 Plasarea prin metode numerice

Problema de plasare poate fi transformată într-o problemă de optimizare numericã. În această secţiune se va prezenta o metodă numită plasare controlatã de forţe, elaboratã de Hanan şi Kurtzberg [146]. Problema de plasare este redusã la soluţionarea unui sistem de ecuaţii liniare pentru a determina locaţiile de echilibru (coordonatele ideale x, y) ale celulelor.

Ideea de bazã a acestei metode este cã celulele interconectate exercitã forţe unele asupra altora. Mãrimea forţei F exercitate de o celulã i asupra altei celule j este proporţionalã cu distanţa care le separã. Aceasta este o analogie cu legea lui Hooke din mecanicã, care se referã la forţele exercitate între douã mase conectate printr-un arc. Dacã masele se aflã la o distanţã d şi constanta arcului este k, forţa 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} distanţa între a şi b. Forţa de atracţie dintre celule este proporţionalã cu produsul $w_{ab} \times d_{ab}$. O celulã i conectatã cu mai multe celule j aflate la distanţe d_{ij} prin conexiuni cu ponderi w_{ij} este atrasã cu o forţã totalã F_i datã de

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

Dacã celula i dintr-un asemenea sistem îşi poate modifica poziţia, această deplasare se va realiza în direcţia forţei F_i pânã când forţa rezultantã asupra celulei va fi zero. Locaţia în care se va deplasa celula este numitã locaţie destinaţie de forţã zero. În ecuaţia (4.41), F_i reprezintã lungimea totalã ponderatã a conexiunilor care pornesc de la celula i. Atunci când toate celulele se deplasează în locaţiile lor de forţã zero, suma pãtratelor distanţelor este minimizatã. Acesta este principiul care stã la baza metodei de plasare controlatã de forţe.

Metoda constã în calcularea forţelor exercitate asupra oricãrei celule date, şi apoi deplasarea acesteia în direcţia forţei rezultante pentru plasarea celulei în locaţia sa de forţã zero. Această locaţie (x_i^o, y_i^o) poate fi determinatã prin egalarea cu zero a componentelor x şi y ale forţelor exercitate asupra celulei, deci

$$\sum_j w_{ij} \cdot (x_j^o - x_i^o) = 0 \tag{4.42}$$

$$\sum_j w_{ij} \cdot (y_j^o - y_i^o) = 0 \tag{4.43}$$

Rezultã pentru x_i^o şi y_i^o:

$$x_i^o = \frac{\sum_j w_{ij} \cdot x_j}{\sum_j w_{ij}} \tag{4.44}$$

$$y_i^o = \frac{\sum_j w_{ij} \cdot y_j}{\sum_j w_{ij}} \tag{4.45}$$

Trebuie evitatã asignarea mai multor celule aceleiaşi locaţii, sau soluţia banalã care asigneazã toate celulele aceleiaşi locaţii (x_i^o, y_i^o).

Procedeul numeric pentru determinarea locaţiilor ideale ale celulelor se poate extinde într-o procedurã de plasare constructivã dupã cum urmeazã. Pornind de la o anumitã plasare iniţialã, este selectatã câte o celulã la un moment dat, şi este calculatã locaţia sa de forţã zero. Procesul poate fi repetat în mod iterativ pentru a îmbunãtãţi soluţia obţinutã. Deciziile care trebuie luate de un asemenea algoritm se referã la ordinea în care sunt selectate celulele şi locaţia în care trebuie plasatã celula în cazul în care locaţia de forţã zero este ocupatã.

Celula care va fi mutatã poate fi selectatã în mod aleator sau utilizând o tehnicã euristicã. Este logicã selectarea celulei *i* pentru care F_i este maximã în configuraøia actualã. Dacã existã restricøii de timp, pot fi selectate mai întâi celulele care se aflã pe cãile critice.

Dacã locaøia de forøã zero a unei celule *p* este ocupatã de o altã celulã *q*, sunt disponibile diferite opøiuni pentru plasarea celulei *p*, dintre care se prezintã urmãtoarele:

1) Se mutã *p* într-o locaøie liberã apropiatã de *q*.
2) Se evalueazã modificarea costului dacã *p* este interschimbat cu *q*. Dacã rezultã o reducere a costului, se efectueazã interschimbarea. Este necesarã calcularea modificãrii costului deoarece este posibil ca celula *q* sã se afle în locaøia sa de forøã zero.
3) Celula *p* este plasatã în locaøia ocupatã, øi este calculatã o nouã locaøie de forøã zero pentru celula *q* deplasatã. Procedura este continuatã pânã când sunt plasate toate celulele.
4) Celula *p* este plasatã în locaøia calculatã øi celula *q* este mutatã într-o locaøie adiacentã. Dacã locaøia adiacentã este ocupatã de o altã celulã *r*, atunci celula *r* este mutatã într-o locaøie adiacentã, øi se continuã astfel pânã când se gãsešte o locaøie liberã.
5) Pentru evitarea acestei probleme se pot calcula locaøiile de forøã zero pentru toate celulele. Se cautã perechi de celule de forma (*p*, *q*) astfel încât locaøia de forøã zero a celulei *p* este poziøia prezentã a celulei *q* øi invers. Se interschimbã apoi celulele *p* øi *q*; au fost gãsite astfel locaøiile de forøã zero ale ambelor celule.

Existã diferite versiuni ale metodei de plasare controlatã de forøe. Se prezintã în Figura 4.12 un algoritm cu îmbunãtãøire iterativã care utilizeazã mutarea de tip 3) pentru rezolvarea conflictului la o locaøie [153].

Algoritmul utilizeazã douã indicatoare pentru fiecare locaøie a suprafeøei de plasare. Un indicator OCUPAT aratã dacã existã o celulã asignatã locaøiei respective. Un indicator BLOCAT aratã starea celulei asignate în mod curent locaøiei. Dacã celula care ocupã locaøia a fost deplasatã cel puøin o datã, indicatorul BLOCAT al locaøiei va fi setat pentru a preveni deplasarea din nou a celulei respective.

În cadrul algoritmului este selectatã câte o celulã, numitã celulã sursã, fiind calculatã conectivitatea øi locaøia destinaøie de forøã zero a acesteia. Selecøia celulelor se bazeazã pe conectivitatea totalã. Dacã locaøia destinaøie este ocupatã, celula care ocupa în prealabil locaøia calculatã va fi selectatã pentru a fi mutatã. Pentru a evita buclele infinite, oricare celulã care este mutatã în locaøia sa destinaøie este blocatã pentru valoarea curentã a contorului de iteraøii *contor_iter*. Buclele infinite pot apare dacã douã celule C_a øi C_b concurează pentru aceeaøi locaøie destinaøie. Odatã ce este selectatã o celulã sursã øi este calculatã locaøia destinaøie de forøã zero a acesteia, sunt posibile patru cazuri: locaøia calculatã poate fi (1) *aceeaøi* cu locaøia iniøialã a celulei sursã, (2) o altã locaøie *liberã*, (3) o locaøie care este *ocupatã* (dar neblocatã), sau (4) o locaøie care este ocupatã øi *blocatã*.

Bucla **while** interioarã a algoritmului este executatã cât timp *end_mutare* este *false*. Dacã locaøia calculatã este aceeaøi cu cea curentã sau este o altã locaøie liberã, indicatorul *end_mutare* este setat la *true*, *contor_aband* este setat la zero øi celula ocupã locaøia calculatã. Este selectatã apoi urmãtoarea celulã sursã în ordinea conectivitãøii øi bucla continuã.

Dacã locaøia calculatã este ocupatã øi nu este blocatã, celula este mutatã în locaøia calculatã, iar celula care ocupa aceastã locaøie este selectatã ca urmãtoarea celulã care va fi mutatã. Indicatorul *end_mutare* este setat la *false*, iar *contor_aband* este setat la zero.

```
Algorithm Plasare_numerică;
begin
        Se calculează conectivitatea totală a fiecărei celule;
        Se sortează celulele în ordine descrescătoare a conectivităţii şi se depun în lista L;
        while (contor_iter < limit_iter) do
                Sursă = următoarea celulă din L;
                Se declară poziţia sursei ca liberă;
                while (end_mutare = false) do
                        Se calculează locaţia destinaţie a sursei şi se rotunjeşte la
                                următorul întreg;
                        case (locaţie_destinaţie) is
                        LIBERĂ:
                                Se mută celula sursă la destinaţie şi se blochează;
                                end_mutare = true;
                                contor_aband = 0;
                        ACEEAŞI CU LOCAŢIA CURENTĂ:
                                end_mutare = true;
                                contor_aband = 0;
                        BLOCATĂ:
                                Se mută celula selectată în locaţia liberă cea mai apropiată;
                                end_mutare = true;
                                contor_aband = contor_aband + 1;
                                if (contor_aband > limit_aband) then
                                        Se deblochează toate locaţiile celulelor;
                                        contor_iter = contor_iter + 1;
                                endif;
                        OCUPATĂ:        /* şi neblocată */
                                Se selectează celula ca fiind următoarea care va fi mutată;
                                Se mută celula sursă la destinaţie şi se blochează;
                                end_mutare = false;
                                contor_aband = 0;
                        endcase;
                endwhile;
        endwhile;
end.
```

Figura 4.12. Exemplu de algoritm de plasare prin metode numerice.

În cazul în care locaţia destinaţie este ocupată şi blocată, indicatorul *end_mutare* este setat la *true*, *contor_aband* este incrementat, iar celula este mutată în cea mai apropiată locaţie liberă. În acest caz, dacă *contor_aband* este mai mic decât *limit_aband*, este selectată următoarea celulă sursă, locaţiile blocate rămân în aceeaşi poziţie, şi iteraţia continuă. Dacă însă *contor_abort* este mai mare decât valoarea prestabilită *limit_aband*, toate locaţiile blocate sunt deblocate, este selectată o altă celulă sursă, contorul de iteraţii este incrementat şi se începe o nouă iteraţie.

4.4.6 Plasarea liniară prin metode spectrale

Plasarea liniară este o problemă fundamentală pentru proiectarea circuitelor VLSI. O plasare liniară de calitate mai bună are ca efect reducerea lungimii conexiunilor, ceea ce reduce probabilitatea ca o conexiune să fie tăiată [139]. Au fost propuse diferite metode care utilizează vectori proprii, atât pentru problema de partiţionare, cât şi pentru plasarea liniară [39] [40] [86] [109].

În literatură s-au efectuat comparaţii între funcţiile obiectiv liniare şi cuadratice. S-a constatat că prin utilizarea unei funcţii liniare se obţine o plasare de calitate mai bună din punct de vedere a lungimii conexiunilor. În [139], utilizarea unei funcţii

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. Cu alte cuvinte, deviaøia standard a lungimii conexiunilor este mai micã în cazul funcøiei cuadratice 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 minimizeazã lungimea pãtratã a conexiunilor în locul lungimii liniare, øi de aceea nu corespunde direct cu scopul plasãrii liniare.

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

Metoda vectorilor proprii asigurã nu numai o soluøie euristicã pentru problema de plasare liniarã, dar furnizeazã øi informaøii globale de conectivitate pentru gruparea componentelor. Aceasta permite reducerea dimensiunii problemei øi poate îmbunãtãøi calitatea soluøiei obøinute prin metodele euristice de partiøionare øi plasare. De asemenea, gruparea are avantaje pentru modelarea conexiunilor multi-pin. În urma grupãrii unui circuit, gradul hipermuchiilor din graf este redus. În consecinøã, modelele grafurilor utilizate pentru aproximarea hipergrafului circuitului vor avea o acurateøe mai mare.

Problema de plasare liniarã a unui circuit modelat printr-un hipergraf $H = (V_H, E_H)$ este de a se plasa componentele reprezentate prin setul de vârfuri V_H în locaøii aflate la distanøe egale (câte o componentã în fiecare locaøie), astfel încât lungimea totalã a conexiunilor sã fie minimã. În general, acest hipergraf este aproximat printr-un graf $G = (V, E)$, unde o hipermuchie este reprezentatã printr-un set de muchii.

Fie $n = |V|$ numãrul de vârfuri øi $m = |E|$ numãrul de muchii ale grafului G. Acest graf poate fi descris printr-o matrice de adiacenøã $A = [a_{ij}]$ cu dimensiunea $n \times n$, unde un element a_{ij} al matricii reprezintã ponderea conexiunii dintre vârfurile i øi j. Dupã transformarea unui hipergraf într-un graf, problema de plasare liniarã se poate formula astfel:

$$\min \sum_{i>j} \sum_{j} a_{ij} |d_i - d_j| \tag{4.46}$$

unde d_i este coordonata locaøiei pentru vârful i în cadrul plasãrii liniare.

Problema de plasare liniarã este NP-completã. Printr-o metodã spectralã, plasarea spectralã continuã, în care restricøia de plasare a vârfurilor în locaøii specifice este eliminatã, se utilizeazã de obicei ca o euristicã pentru soluøionarea problemei de plasare liniarã.

Fiind dat graful ponderat $G = (V, E)$, reprezentat prin matricea de adiacenøã $A = [a_{ij}]$, matricea Q, numitã Laplacian al lui G, este definitã astfel:

$$Q = \begin{cases} \sum_{j=1}^{n} a_{ij} & \text{dacã } i = j \\ -a_{ij} & \text{în caz contrar} \end{cases} \tag{4.47}$$

Cu metoda vectorilor proprii, problema de plasare liniarã continuã este formulatã ca o problemã de programare cuadraticã, astfel:

$$\min \sum_{i>j} \sum_{j} a_{ij} (x_i - x_j)^2 = X^T Q X \tag{4.48}$$

astfel încât

$$I^T X = 0, X^T X = 1$$

unde x_i este coordonata vârfului *i* în cadrul plasării liniare continue.

Prin calcularea celei mai mici valori proprii diferite de zero a Laplacianului *Q* şi a vectorului propriu corespunzător *X*, se obţine o soluţie non-trivială a problemei de programare cuadraticã de mai sus, iar soluţia euristicã a problemei de plasare liniarã se obţine prin interpretarea vectorului propriu ca o ordonare liniarã a vârfurilor *V*. Cu o asemenea metodã, funcţia obiectiv care este minimizatã este lungimea pãtratã a conexiunilor.

În [139] a fost utilizatã urmãtoarea funcţie obiectiv liniarã pentru rezolvarea problemei de plasare liniarã continuã:

$$\min \sum_{i>j} \sum_{j} a_{ij} |x_i - x_j| \tag{4.49}$$

astfel încât

$$\sum_i x_i = f$$

Prin utilizarea acestei funcţii, s-a obţinut o calitate ridicatã a plasării şi partiţionării.

O asemenea funcţie obiectiv liniarã poate fi rescrisã ca o funcţie cuadraticã prin modificarea a_{ij} cu distanţa $|x_i - x_j|$ [109] [139]:

$$\sum_{i>j} \sum_{j} a_{ij} |x_i - x_j| = \sum_{i>j} \sum_{j} a'_{ij} (x_i - x_j)^2 \tag{4.50}$$

unde

$$a'_{ij} = \frac{a_{ij}}{|x_i - x_j|}$$

Astfel, prin îmbunãtãţire iterativã, metoda programării cuadratice poate fi utilizatã pentru rezolvarea problemei de plasare liniarã cu funcţia de obiectiv liniarã.

Pentru problema de plasare liniarã, funcţia obiectiv liniarã permite obţinerea unei calitãţi mai ridicate a soluţiei decât funcţia obiectiv cuadraticã; aceasta deoarece funcţia liniarã reprezintã o mãsurã mai exactã pentru problema de plasare liniarã. Pe de altã parte, funcţia obiectiv cuadraticã are avantajele aminitite anterior.

Deoarece problema de plasare liniarã este soluţionatã în mod euristic prin interpretarea vectorului propriu ca ordonare a vârfurilor, cu cât vârfurile sunt plasate mai dispersat, cu atât vor fi introduse mai puţine erori numerice în cadrul plasării. De aceea, soluţia continuã a plasării liniare trebuie sã fie suficient de dispersatã pentru a putea fi interpretatã.

Pe baza acestor observaţii, Li *et al.* [109] au propus o funcţie obiectiv de ordin α pentru problema de plasare liniarã continuã, astfel:

$$\min \sum_{i>j} \sum_{j} a_{ij} |x_i - x_j|^{\alpha} = \sum_{i>j} \sum_{j} \frac{a_{ij}}{|x_i - x_j|^{2-\alpha}} (x_i - x_j)^2 \tag{4.51}$$

astfel încât

$$\sum_i x_i = f$$

unde $1.0 \le \alpha \le 2.0$. Atunci când $\alpha = 1.0$, funcţia de ordin α devine funcţia liniarã, iar atunci când $\alpha = 2.0$, funcţia de ordin α devine funcţia cuadraticã. Cu aceastã funcţie obiectiv, se mãreşte dispersia soluţiilor, pãstrându-se în acelaşi timp o mãsurã suficient de exactã pentru lungimea conexiunilor. Plasarea liniarã continuã cu funcţia obiectiv de ordin α poate fi soluţionatã în mod iterativ, şi o asemenea metodã iterativã este

convergentã, lungimea conexiunilor, definitã în ecuaøia (4.51), fiind monoton descrescãtoare [109].

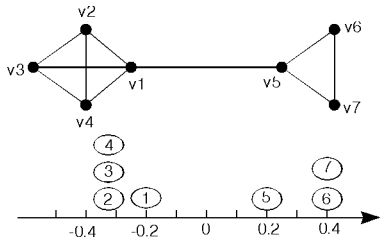

Figura 4.13. Exemplu de graf şi plasarea liniarã continuã prin metoda vectorilor proprii.

Soluøia problemei de plasare liniarã continuã nu reprezintã numai o euristicã pentru soluøionarea problemei de plasare liniarã, dar şi o euristicã pentru problema de grupare a vârfurilor din graful circuitului. În Figura 4.13 se prezintã un graf simplu şi plasarea liniarã continuã obøinutã prin metoda vectorilor proprii. În acest exemplu, existã 7 vârfuri $V = \{x_i \mid i = 1, \ldots, 7\}$ şi 10 conexiuni. Vectorul propriu corespunzãtor celei mai mici valori proprii diferite de zero $\lambda_1 = 0.3983$ pentru acest exemplu este $X = \{x_1 = -0.2142, x_2 = x_3 = x_4 = -0.3560, x_5 = 0.2965, x_6 = x_7 = 0.4928\}$.

Din aceastã plasare liniarã continuã, vârfurile V se pot divide în douã grupuri. Primul grup cuprinde vârfurile $(v_i \mid i = 1, \ldots, 4)$, iar al doilea grup cuprinde vârfurile $(v_i \mid i = 5, 6, 7)$. Aceasta deoarece vârfurile din fiecare grup sunt plasate foarte apropiat, în timp ce distanøa dintre cele douã grupuri este relativ mare. Conectivitatea din cadrul unui grup este deci mult mai ridicatã decât conectivitatea între grupuri. Astfel, este naturalã utilizarea rezultatelor plasãrii liniare continue ca şi informaøii de conectivitate globalã într-un proces de grupare.

Se prezintã în continuare un algoritm de plasare liniarã care combinã metoda spectralã cu gruparea circuitului [109]. Acest algoritm constã din trei faze: 1) prin utilizarea unei metode de grupare de jos în sus, se formeazã grupuri de noduri cu conectivitate ridicatã; 2) se executã plasarea liniarã a circuitului grupat; 3) se executã plasarea liniarã a circuitului original. În ultima fazã, se expandeazã mai întâi grupurile şi se executã plasarea liniarã pentru nodurile fiecãrui grup. Apoi, se îmbunãtãøeşte calitatea plasãrii liniare printr-un algoritm care utilizeazã metoda fluxului maxim şi a tãieturii minime. Algoritmul este prezentat în Figura 4.14.

Algoritmul de grupare a nodurilor circuitului foloseşte proprietatea de grupare a metodei vectorilor proprii ca şi informaøie de conectivitate globalã. Scopul principal al grupãrii este de a reduce dimensiunea circuitului fãrã a afecta calitatea partiøionãrii şi a plasãrii. Acest algoritm iniøializeazã fiecare grup cu un singur nod al circuitului original. Se fuzioneazã apoi în mod iterativ perechi de grupuri care sunt puternic conectate, pânã când se reduce dimensiunea circuitului grupat pânã la o valoare predefinitã.

Se utilizeazã urmãtoarea metricã pentru a mãsura conectivitatea unei perechi de grupuri i şi j:

$$C = \frac{\sum_k c_{ij}^k}{W_i \times W_j} \times g(i,j) \tag{4.52}$$

unde c_{ij}^k reprezintã contribuøia conexiunii cu numãrul de ordine k la conectivitatea circuitului. Dacã grupurile i şi j sunt conectate prin conexiunea k, aceastã contribuøie este

$$c_{ij}^{k} = \frac{1}{\dim(conexiune_k) - 1},$$

iar în caz contrar, $c_{ij}^{k} = 0$. W_i indică dimensiunea grupului i; dacă dimensiunea grupului este mare, este descurajată participarea acesteia la operaţii suplimentare de grupare. În fine, $g(i, j)$ este ponderea rezultată din plasarea liniară continuă prin metoda vectorilor proprii, având rolul unei informaţii globale de conectivitate.

Algorithm *Plasare_Liniară_Spectrală*;
begin
1. Se execută un algoritm de grupare a nodurilor circuitului;
   Fie $\Theta = \{C_i \mid i = 1, \ldots, K\}$ setul de grupuri;
2. Se construieşte noul circuit grupat pentru Θ;
   Se execută plasarea liniară cu funcţia obiectiv de ordin α;
3. Fie $O = (C_1, \ldots, C_K)$ ordonarea grupurilor;
   $i = 2$;
   while ($i \le K$ -1) **do**
      Se construieşte pentru C_i o reţea rezistivă (s, v_{ji}, …, v_{jl}, t),
         unde v_{ji} ($i = 1 \ldots l$) este o componentă a lui C_i;
      Se execută plasarea liniară a acestei reţele rezistive;
      Se înlocuieşte C_i cu ordonarea (v_{i1}, …, v_{il}) în O;
      $i = i + 1$;
   endwhile;
end.

Figura 4.14. Algoritmul de plasare liniară prin metoda spectralã.

În ultima fază a algoritmului se utilizează conceptul reţelelor rezistive pentru rezolvarea problemei de plasare. Prin analogie cu un circuit electric, unde conductanţa între nodul i şi nodul j este egală cu $-q_{ij}$, elementul din matricea Q (Laplacian) definită în ecuaţia (4.47), s-a demonstrat că problema de plasare este echivalentă cu cea a alegerii tensiunilor circuitului electric astfel încât puterea disipată să fie minimă. Tensiunile nodurilor circuitului electric reprezintă o analogie cu coordonatele nodurilor pentru problema de plasare. Spre deosebire de metoda vectorilor proprii, reţeaua rezistivă poate include specificaţiile celulelor de I/E. Astfel, prin modelarea celulelor de I/E ca surse de tensiune fixă, se pot fixa anumite noduri la coordonate specifice, fiind necesară determinarea coordonatelor celorlalte noduri.

Metoda reţelelor rezistive poate fi utilizată pentru determinarea coordonatelor nodurilor dintr-un grup, rezultând decomprimarea circuitului grupat. Fiind dată ordonarea grupurilor $O = (C_1, C_2, \ldots, C_K)$, atunci când se realizează decomprimarea grupului C_i, se încearcă determinarea coordonatelor nodurilor din C_i în timp ce ordonarea grupurilor nu este schimbată. Prin metoda reţelelor rezistive, se comprimă toate grupurile de la stânga grupului C_i din O într-o celulă de I/E s cu tensiunea 0, se înlocuieşte grupul C_i cu componentele sale, şi se comprimă toate grupurile de la dreapta grupului C_i din O într-o celulă de I/E t cu tensiunea 1. Apoi se determină coordonatele nodurilor din C_i pe baza tensiunilor acestei reţele rezistive comprimate. Pentru aceasta se utilizează următoarea teoremă:

> **Teorema 4.4.6.** [109] În reţeaua rezistivă comprimată de sus, tensiunea unui nod mobil este între 0 şi 1.

Pe baza acestei teoreme, nodurile din C_i vor fi plasate între două noduri comprimate s şi t. Astfel, procesul de decomprimare nu va modifica ordonarea grupurilor definită de O.

După decomprimarea grupului $C_i \mid i = 2, \ldots, K$-1 în ordonarea O, se obţine o soluţie a problemei de plasare liniară pentru circuitul original. Această metodă de gru-

pare şi decomprimare permite obøinerea unei plasări de calitate mai ridicatã din punct de vedere a lungimii conexiunilor decât metoda vectorilor proprii cu o funcøie obiectiv liniarã.

4.4.7 Alte metode iterative

4.4.7.1 Plasarea prin asignare cuadratică

Problema de asignare cudraticã poate fi definitã astfel: Fiind datã o matrice a costurilor **C** = [c_{ij}], o matrice a distanøelor **D** = [d_{ij}], şi o permutare p a primilor n întregi, sã se determine minimul expresiei

$$Q = \sum_{i,j} c_{ij} d_{p(i)p(j)} \tag{4.53}$$

pentru toate permutãrile p.

Una din interpretãrile posibile ale acestei formulãri este urmãtoarea: Fiind date n persoane (module) şi o matrice **C** = [c_{ij}], unde c_{ij} este o mãsurã a afinitãøii (capacitãøii) între persoanele i şi j (modulele i şi j), n birouri posibile (locaøii) pentru aceste persoane (module), şi o matrice **D** = [d_{st}], unde d_{st} este distanøa mãsuratã cu o anumitã metricã între birourile (locaøiile) s şi t. Dacã i este asignat biroului (locaøiei) $p(i)$, iar j este asignat biroului (locaøiei) $p(j)$, costul acestei asignări este $c_{ij}d_{p(i)p(j)}$. Astfel, costul total pentru fiecare i şi j este Q, definit în ecuaøia (4.52). Se presupune cã afinitatea reprezintã mãrimea comunicării directe. Obiectivul este de a minimiza distanøa totalã parcursã (lungimea totalã de rutare) de toate persoanele.

Deoarece existã n ! permutări pentru n întregi, existã n ! posibilitãøi de asignare a modulelor la cele n locaøii.

Comparând problema de asignare cuadraticã cu problema generalã de plasare, se observã cã problema de asignare cuadraticã este un caz special al problemei de plasare, prin presupunerea cã toate seturile de semnale conøin numai douã module. În cazul problemei de asignare cuadraticã, se considerã numai perechi de puncte, în timp ce în cazul problemei de plasare se considerã un set de puncte sau seturi de semnale. Totuşi, tehnicile pentru rezolvarea problemelor de asignare cuadraticã pot fi utilizate pentru problemele de plasare dacã se utilizează urmãtoarea transformare: se seteazã c_{ij} egal cu suma ponderilor seturilor de semnale comune cu modulele i şi j pentru $i \neq j$, cu $c_{ii} = 0$, şi se alege matricea **D** = [d_{ij}] ca fiind una mãsuratã într-o anumitã metricã a problemei originale de plasare.

O soluøie optimã pentru problema asociatã de asignare cuadraticã nu este în mod necesar optimã pentru plasarea originalã datoritã transformării matematice utilizate. De aceea, transformarea utilizatã afectează în mare mãsurã rezultatele.

4.4.7.2 Plasarea pe baza optimizării reţelelor rezistive

Ideea de utilizare a analogiei cu reøelele rezistive pentru soluøionarea problemelor de plasare a fost introdusã mai întâi de Charney şi Plato, iar apoi de Cheng şi Kuh, care au prezentat o metodã sistematicã pentru soluøionarea problemei de plasare prin aceastã analogie [189].

Funcøia obiectiv este datã de:

$$\Phi(x,y) = \frac{1}{2}\sum_{i,j=1}^{n} c_{ij} d_{ij}^2 = \frac{1}{2}\sum_{i,j=1}^{n} c_{ij}[(x_i - x_j)^2 + (y_i - y_j)^2] \tag{4.54}$$

unde d_{ij} este distanţa euclidiană între modulele *i* şi *j*, iar *x* şi *y* reprezintă lungimile conexiunilor măsurate în direcţia *x*, respectiv *y*. Într-un mod mai concis, ecuaţia de sus se poate scrie ca:

$$\Phi = \mathbf{X}^{\mathrm{T}}\mathbf{Q}\mathbf{X} + \mathbf{Y}^{\mathrm{T}}\mathbf{Q}\mathbf{Y} \tag{4.55}$$

unde **Q** = **G** - **C** este o matrice simetrică cu dimensiunea $n \times n$, **C** = $[c_{ij}]$ este matricea de capacitate, iar **G** este o matrice diagonală cu $g_{ij} = \sum_{j=1}^{n} c_{ij}$ [hal]. **X** şi **Y** sunt vectorii distanţelor măsurate în direcţia *x*, respectiv *y*. Funcţia obiectiv corespunde puterii *P* disipate într-o reţea rezistivă:

$$P = \mathbf{V}_n{}^{\mathrm{T}}\mathbf{Y}_n\mathbf{V}_n \tag{4.56}$$

unde $\mathbf{V}_n$ este vectorul reprezentând tensiunile nodurilor, iar $\mathbf{Y}_n$ este matricea simetrică a admitanţelor.

Astfel, problema de plasare a modulelor poate fi formulată ca o problemă de optimizare liniară a unei reţele rezistive, dacă se efectuează următoarea transformare: modulele fixe corespund nodurilor cu tensiuni constante, iar modulele care pot fi mutate corespund nodurilor ale căror tensiuni trebuie determinate.

4.4.7.3 Plasarea prin algoritmul Steinberg

Algoritmul Steinberg foloseşte un set independent de module în timpul fiecărei iteraţii, utilizând o procedură optimă pentru repoziţionarea acelui set [189] Ideea de bază este că atunci când este ales un set independent, modulele din acel set sunt eliminate de pe suprafaţa de plasare, şi se calculează costul plasării fiecăruia din aceste module în fiecare locaţie disponibilă, obţinându-se o soluţie optimă. Deoarece aceste costuri sunt independente de locaţiile celorlalte module din setul independent, poziţionarea acestor module devine o problemă de asignare liniară. După se execută iteraţiile pentru toate seturile independente care sunt generate, ciclul se termină şi începe următorul ciclu.

În general, procesul de construire a unui set independent începe prin alegerea aleatoare a unui modul, şi continuă prin adăugarea la acest set doar a acelor module care nu sunt conectate cu membrii setului. Acest proces continuă până când s-a format un set cu dimensiunea doritã sau nu mai pot fi adăugate alte module la set. Dimensiunea setului independent care va fi utilizat depinde în mare măsură de puterea de calcul disponibilă pentru problema de plasare. Este de preferat utilizarea seturilor independente maxime ori de câte ori este posibil pentru a se asigura flexibilitatea maximă în cadrul problemei de plasare.

O îmbunătăţire a algoritmului Steinberg a fost sugeratã de Rutman, în scopul obţinerii unor rezultate mai bune şi pentru a obţine convergenţa mai rapidă a algoritmului.

4.4.7.4 Plasarea prin metoda spațiului grafurilor

Unii autori au încercat îmbunătăţirea plasării utilizând metode bazate pe teoria grafurilor. Una din aceste metode este numită metoda spaţiului grafurilor, în care relaţiile de conexiune între modulele din cadrul unui circuit electric sunt reprezentate printr-un hipergraf. O muchie a unui hipergraf poate conecta orice număr de noduri, în timp ce o muchie a unui graf obişnuit nu poate conecta mai mult de două module. Prin utilizarea acestei metode, nodurile (reprezentând module) sunt mapate în spaţiul grafului astfel încât distanţa între nodurile din spaţiu reflectă ponderea (numărul de conexiuni) unei muchii între nodurile hipergrafului original. Se minimizează apoi lungimea totală de rutare prin aplicarea tehnicii primei asignări propuse de Hung şi Rom [189].

O altã metodã care utilizeazã un *digraf relaxat* a fost propusã de Ciesielski şi Kinnen pentru soluøionarea problemei de plasare bidimensionalã cu scopul de a se minimiza spaøiul utilizat de module cu dimensiuni arbitrare.

4.4.7.5 Plasarea bazată pe operaţii cu linii şi coloane

Se considerã o placã bidimensionalã pe care se plasează module în poziøiile definite prin coordonatele *x* şi *y*. Fiecare modul va aparøine unei linii şi coloane. Un algoritm bazat pe operaøii cu linii şi coloane constã din doi paşi. Întâi, se determinã o plasare optimã localã pentru module într-o linie (coloanã), şi apoi se obøine plasarea îmbunătãøitã între două linii (coloane). Procesul este iterat şi se terminã atunci când se obøine un rezultat satisfãcãtor sau s-a epuizat un anumit timp de calcul.

Problema de determinare a plasării optime într-o linie (coloanã) este definitã astfel: Fiind date *n* module, *n* locaøii într-o linie, şi o matrice de capacitate **C** = $[c_{ij}]$, unde c_{ij} este ponderea între modulele *i* şi *j*, trebuie determinatã lungimea minimã de rutare prin asignarea unui singur modul fiecãrei locaøii. Numãrul de soluøii existente este *n* !, astfel cã pentru un numãr mare de module nu se pot examina toate soluøiile posibile. Pentru problema asociatã de asignare, metoda elaboratã de Munkres poate gãsi soluøia într-un timp rezonabil, în cazul unui numãr de module de ordinul 200. Un algoritm euristic a fost elaborat de Wang şi Chen pe baza noøiunii de arbore maxim de acoperire într-un graf general.

4.5 Metode neconvenţionale de plasare

În această secøiune sunt prezentate unele metode neconvenøionale utilizate pentru rezolvarea problemei de plasare. În secøiunea 4.5.1 este prezentatã implementarea paralelã a unui algoritm de plasare prin metoda cãlirii simulate. În secøiunea 4.5.2 se prezintã utilizarea reøelelor neuronale artificiale pentru plasare.

4.5.1 Plasarea prin algoritmi paraleli

În această secøiune se prezintã implementarea paralelã a unui algoritm de plasare care se bazează pe metoda de cãlire simulatã. 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 să conducă la o configuraøie fezabilã unicã. Pentru a reduce timpul de calcul, se pot utiliza două 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 implementări paralele ale metodei de cãlire simulatã [104] [141]. Diferenøele constau în principal în următoarele aspecte:

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

Pentru implementarea paralelă a problemei de plasare au fost sugerate diferite soluţii. Metoda utilizată de Casotto *et al.* [37] constă în partiţionarea setului de celule care trebuie plasate într-un număr de subseturi egal cu numărul procesoarelor disponibile; fiecare subset fiind asignat unui anumit procesor. Procesoarele funcţionează asincron cât timp mutările apar într-un set dat de celule. Sunt permise interschimbările celulelor între diferite seturi, dar aceasta presupune suspendarea unuia din cele două procesoare implicate. Mutările din cadrul unui set de celule sunt translaţii, rotaţii sau interschimbări între celule. Deoarece procesoarele funcţionează asincron, niciunul din procesoare nu cunoaşte exact care este configuraţia curentă. Această metodă a fost implementată pe un calculator Sequent Balance 8000 cu maxim 8 procesoare. Casotto a experimentat de asemenea paralelizarea masivă a algoritmului de călire simulată pe calculatorul Connection Machine.

Mallela şi Grover [119] au utilizat de asemenea grupuri de celule pentru a reduce numărul celulelor care trebuie plasate în cadrul fiecărei subprobleme. Plasarea celulelor în fiecare grup implică un spaţiu de căutare redus, deci un timp de calcul redus, fiind posibilă evaluarea în paralel.a fiecărui grup.

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 că două procesoare nu pot muta aceleaşi celule simultan; astfel, nu există conflict între procesoare şi configuraţia finală este întotdeauna validă [142]. Atunci când o perturbaţie este acceptată, configuraţia celulelor este actualizată, indiferent de mutările care se calculează. La temperaturi joase, când rata de acceptare este redusă, nu există o diferenţă importantă faţă de algoritmul secvenţial. La temperaturi mai înalte, comportarea metodei paralele diferă în mod semnificativ de cea a metodei secvenţiale. Măsurătorile au fost efectuate pe un mediu de emulare care permite simularea unui sistem multiprocesor cu memorie partajată cu până la 64 de procesoare.

O metodă diferită a fost utilizată de Kravitz şi Rutenbar [104], care au introdus noţiunea de subset de mutări serializabile. Un set de mutări este serializabil dacă mutările nu interacţionează unele cu altele. Dacă mutările unui asemenea subset sunt evaluate în paralel, rezultatul este acelaşi ca şi în cazul evaluării secvenţiale. Determinarea acestor subseturi este însă din ce în ce mai dificilă şi consumatoare de timp pe măsură ce numărul de procesoare creşte. Soluţia care a fost sugerată la această problemă este considerarea “celui mai simplu subset serializabil”, unde este acceptată o singură mutare acceptabilă, în timp ce toate mutările respinse sunt contorizate. La temperaturi înalte, algoritmul paralel este foarte diferit de cel secvenţial, deoarece rata de acceptare nu este aceeaşi.

Roussel-Ragot şi Dreyfus [142] au sugerat o implementare paralelă a algoritmului de călire simulată care, pe de o parte, este independentă de problemă, iar pe de altă parte, are aceleaşi proprietăţi de convergenţă ca şi algoritmul serial. Aceştia au utilizat două moduri de paralelizare, în funcţie de valoarea temperaturii, şi au elaborat modele statistice care pot estima creşterea de vitezã pentru orice problemă, în funcţie de rata de acceptare şi numărul de procesoare.

Rata de acceptare $\chi(T)$ este raportul între numărul de mutări acceptate şi numărul de mutări încercate, pentru o anumită temperaturã. Una din caracteristicile algoritmului de călire simulată este reducerea ratei $\chi(T)$ pe măsură ce temperatura scade. Cele mai multe metode paralele de călire simulată utilizează acest fapt. În regimul de temperaturi joase, când rata de acceptare χ este redusă, se pot utiliza K procesoare astfel încât $K < 1/\chi$. Deci, va fi acceptată cel mult o mutare în timp ce sunt evaluate K mutări, astfel încât este de aşteptat ca timpul de calcul în modul paralel să fie mai redus decât timpul de calcul în modul secvenţial cu un factor de ordinul K.

Pe lângă timpul de calcul, convergenţa algoritmului este de importanţă deosebită în scopul obţinerii unei soluţii valide (sau a uneia optime). Unele metode de paralelizare pot afecta convergenţa algoritmului, în unele cazuri această convergenţă

fiind chiar imposibilã. Au fost efectuate diferite studii teoretice ale algoritmului secvenøial, dar pentru algoritmul paralel de cãlire simulatã teoria este mult mai puøin dezvoltatã. De aceea, este avantajoasã utilizarea unei metode paralele care respectã condiøiile de convergenøã ale algoritmului secvenøial.

În cazul metodei propuse în [142], pentru ca algoritmul paralel sã aibã o convergenøã similarã cu cel secvenøial, se urmãreşte generarea unui lanø Markov ale cãrui stãri au aceeaşi distribuøie a probabilitãøii ca şi lanøul secvenøial, dacã toøi ceilaløi parametri sunt aceeaşi. În funcøie de rata de acceptare $\chi(T)$, se considerã douã regimuri:

- Un regim al temperaturilor joase: Dacã $\chi(T) < 1/K$, va fi acceptatã cel mult o mutare din K. Procesoarele încearcã mutãri pe cont propriu, în mod asincron, în paralel, pânã când unul din cele K procesoare acceptã o mutare. Dacã este gãsitã o mutare acceptatã, procesoarele sunt sincronizate, memoriile lor sunt actualizate cu noua configuraøie, şi începe urmãtorul pas de evaluare.
- Un regim al temperaturilor înalte ($\chi(T) > 1/K$): În acest regim, fiecare procesor poate evalua o singurã mutare, şi aşteaptã pânã când toate celelalte procesoarele îşi terminã evaluarea. Apoi, se alege în mod aleator una din mutãrile acceptate, memoriile procesoarelor sunt actualizate cu noua configuraøie, şi începe urmãtorul pas de evaluare.

În plus pe lângã cele K procesoare "slave" menøionate, metoda necesitã un procesor "master", care monitorizeazã procesul de cãlire, alege mutarea acceptatã în regimul temperaturilor înalte, actualizeazã memoria fiecãrui procesor, şi øine evidenøa statisticilor.

Pentru elaborarea unui model statistic al celor douã regimuri de temperaturã, se noteazã cu:

L_a numãrul maxim de mutãri acceptate la o temperaturã datã;

L_t numãrul maxim de încercãri la o temperaturã datã;

τ_0 timpul mediu de calcul necesar pentru evaluarea unei mutãri în modul secvenøial;

τ_r timpul mediu necesar comunicaøiei cu cele K procesoare "slave" şi sincronizãrii acestora.

Semnificaøia pentru τ_0 este clarã numai dacã se utilizeazã un singur tip de mutare elementarã; în caz contrar, τ_0 reprezintã valoarea medie a timpului de calcul pentru diferitele tipuri de mutãri care pot apare în timpul cãlirii simulate. În ambele cazuri, τ_0 poate fi estimat ca durata unui pas de temperaturã împãrøitã cu numãrul de mutãri încercate. Dacã perturbaøiile variazã cu temperatura (de exemplu, dacã interschimbãrile blocurilor apar în principal la temperaturi înalte şi translaøiile apar mai ales la temperaturi joase), valoarea τ_0 va fi variabilã cu temperatura.

Temperatura este micşoratã fie când numãrul de mutãri acceptate la temperatura curentã ajunge la L_a, fie când numãrul de mutãri încercate la temperatura curentã ajunge la L_t, indiferent care din aceste limite este atinsã prima.

A. Regimul temperaturilor înalte. În acest regim, fiecare procesor evalueazã o mutare, şi toate procesoarele sunt sincronizate la sfârşitul fiecãrei evaluãri. Timpul mediu necesar pentru execuøia unei evaluãri de cãtre cele K procesoare va fi $\tau_0 + \tau_r$. De notat cã τ_r øine cont de faptul cã durata mutãrilor poate fi diferitã, astfel încât timpul necesar terminãrii unei evaluãri paralele este egal cu durata celei mai lungi mutãri încercate.

Deoarece lungimea lanţului Markov depinde de numărul de mutări acceptate şi/sau de numărul de mutări încercate, trebuie evaluate mai întâi aceste cantităţi. Se presupune că, după o evaluare paralelă a *K* mutări, *r* mutări din *K* sunt respinse. Deci, *K* - *r* mutări vor fi acceptabile, dar numai una din ele va fi acceptată în lanţul Markov. Raportul dintre numărul de mutări acceptate şi numărul de mutări încercate, în modul secvenţial, este (*K* - *r*) / *K*. În modul paralel, o singură mutare va fi acceptată.

Se pune problema construirii unui lanţ cu aceeaşi rată de acceptare ca şi în modul secvenţial. Pentru aceasta, se poate proceda în felul următor: Se numerotează procesoarele într-o ordine arbitrară, de la 1 la *K*. Se notează cu *n* numărul primului procesor din listă care acceptă o mutare, şi se construieşte lanţul Markov cu primele *n* mutări din listă. În modul secvenţial, un lanţ Markov cu *n* - 1 mutări respinse, urmat de o mutare acceptată, ar fi avut aceeaşi probabilitate. Se poate arăta că valoarea medie pentru *n* este egală cu $n^* = (K + 1) / (K - r + 1)$ [142].

Dacă toate cele *K* procesoare resping mutările lor încercate, numărul primului procesor nu poate fi determinat utilizând această metodă, dar deoarece nu există mutare acceptată, numărul mutărilor încercate care trebuie luate în considerare este *K*.

Deci, atunci când se execută în paralel *K* evaluări, se poate scrie că:

- Dacă cel puţin o mutare a fost acceptată, se alege în mod aleator una din aceste mutări, şi se consideră că au fost încercate un număr de mutări egal cu $n^* = (K + 1) / (K - r + 1)$.
- Dacă nici o mutare nu a fost acceptată, se consideră că au fost încercate *K* mutări.

Aceasta permite o estimare corectă a numărului de mutări încercate efectiv, deoarece se poate arăta că, în medie, raportul dintre numărul de mutări acceptate şi numărul efectiv de mutări încercate în modul paralel este egal cu χ, rata de acceptare în modul secvenţial [142]. Astfel se obţine aceeaşi convergenţă în modul paralel ca şi în modul secvenţial, deoarece algoritmul paralel are aceeaşi matrice de probabilitate a tranziţiilor ca şi cel secvenţial.

Se evaluează în continuare numărul total efectiv de mutări încercate N_t^*, şi numărul total de mutări acceptate N_a^*, după ce au fost executate *N* evaluări paralele pentru *K* mutări.

Numărul mediu efectiv de încercări este dat de [142]

$$N_t^* = N \cdot \left[(1-\chi)^K \cdot K + \sum_{i=1}^{K} \binom{K}{i} \cdot \chi^i (1-\chi)^{K-i} \frac{K+1}{i+1} \right] \tag{4.57}$$

sau, în mod echivalent,

$$N_t^* = N \cdot \frac{1-(1-\chi)^K}{\chi} \tag{4.58}$$

Limita L_t a mutărilor încercate este atinsă după un număr N_t de evaluări paralele a *K* mutări, care este dat de

$$N_t = L_t \cdot \frac{\chi}{1-(1-\chi)^K} \tag{4.59}$$

Numărul N_a^* de mutări acceptate este egal cu numărul de evaluări paralele a *K* mutări care conduc la cel puţin o mutare acceptabilă, deci

$$N_a^* = N[1-(1-\chi)^K] \tag{4.60}$$

Astfel, limita L_a a mutãrilor acceptate este atinsã dupã un numãr N_a de evaluãri paralele a K mutãri, care este dat de

$$N_a = \frac{L_a}{1-(1-\chi)^K} \tag{4.61}$$

Deci, în regimul temperaturilor înalte, numãrul de evaluãri paralele la o anumitã temperaturã este

$$N_p = \min (N_t, N_a) \tag{4.62}$$

Timpul de calcul corespunzãtor este

$$t_p = N_p (\tau_0 + \tau_r) \tag{4.63}$$

În modul serial, timpul de calcul este

$t_s = \tau_0 \cdot L_a / \chi$, dacã limita L_a este atinsã prima

$t_s = \tau_0 \cdot L_t$, dacã limita L_t este atinsã prima

Astfel, oricare din limitele L_a sau L_t este atinsã prima

$$\frac{t_p}{t_s} = \frac{\chi}{1-(1-\chi)^K}\left(1+\frac{\tau_r}{\tau_0}\right) \tag{4.64}$$

De notat cã lim $(t_p/t_s) = 1 + \tau_r / \tau_0$ atunci când $\chi \to 1$, iar lim $(t_p/t_s) = 1/K \cdot (1 + \tau_r / \tau_0)$ atunci când $\chi \to 0$.

La temperaturi înalte, eficienøa este scãzutã deoarece este respins un numãr mic de mutãri, rata de acceptare fiind ridicatã. La temperaturi joase, timpul de calcul este divizat de numãrul de procesoare, dacã timpul de comunicaøie τ_r este redus comparativ cu τ_0.

Valoarea medie pentru τ_0 este cunoscutã de la algoritmul secvenøial. Determinarea valorii τ_r nu este simplã øi depinde de problemã. Dacã τ_0 este constant, τ_r poate fi aproximat cu timpul de comunicaøie multiplicat cu K: dacã cele K calcule în paralel se terminã în acelaøi timp, vor fi necesare K comunicaøii succesive pentru ca procesorul "master" sã cunoascã toate rezultatele øi sã relanseze operaøiile executate de procesoarele "slave".

B. Regimul temperaturilor joase. În acest regim, fiecare procesor evaluează mutãrile în mod independent pânã când unul din cele K procesoare acceptã o mutare. La sfârøitul fiecãrei evaluãri individuale, procesoarele transmit în mod asincron procesorului "master" rezultatul evaluãrii. Dacã nici o mutare nu a fost acceptatã de la comunicaøia precedentã, se încearcã o altã mutare. Dacã a fost acceptatã o mutare, memoriile procesoarelor "slave" sunt actualizate øi procesoarele sunt sincronizate. În acest mod, toate mutãrile respinse sunt contorizate ca paøi cãtre realizarea stãrii de echilibru.

Pentru a modela comportarea la temperatura joasã, este necesarã evaluarea numãrului de mutãri necesare pentru ca o mutare sã fie acceptatã. Pentru aceasta, se poate estima timpul necesar pentru mutãrile individuale pe baza numãrului de mutãri în cazul algoritmului secvenøial. În medie, o configuraøie este acceptatã atunci când sunt evaluate $1/\chi$ mutãri. În modul serial, procesul a avansat deci cu $1/\chi$ paøi cãtre starea de echilibru. Deoarece se doreøte obøinerea unei configuraøii fezabile a sistemului, dacã o altã mutare este acceptatã de una din celelalte K - 1 procesoare, aceasta nu este luatã în considerare. Perturbaøiile sunt alese în mod aleator, øi dacã, de exemplu, unul din blocuri este interschimbat în douã mutãri diferite, configuraøia rezultatã nu este validã, deoarece blocul ar trebui plasat în douã locaøii diferite. Atunci când procesoarele sunt sincronizate, au fost evaluate $1/\chi + K$ - 1 mutãri øi, în medie,

au fost acceptate $(1/\chi + K - 1) \cdot \chi$ mutări. Deoarece sunt eliminate toate mutările acceptate, cu excepţia uneia, se contorizează în lanţul Markov doar un număr de paşi egal cu:

$$(1/\chi + K - 1) - (1/\chi + K - 1) \cdot \chi + 1 = 1/\chi + (K - 1) \cdot (1 - \chi)$$

Dacă τ_m este timpul necesar pentru a obţine o mutare acceptată în paralel, comportarea în regimul temperaturilor joase poate fi modelat astfel:

i. Dacă limita L_a este atinsă prima:

$$t_p = L_a \cdot \tau_m \tag{4.65}$$

$$t_s = L_a / \chi \cdot \tau_0 \tag{4.66}$$

deci

$$\frac{t_p}{t_s} = \chi \frac{\tau_m}{\tau_0} \tag{4.67}$$

ii. Dacă limita L_t este atinsă prima:

$$t_p = \frac{L_t}{1/\chi + (K-1)\ (1-\chi)} \tau_m \tag{4.68}$$

$$t_s = L_t \cdot \tau_0 \tag{4.69}$$

deci

$$\frac{t_p}{t_s} = \frac{1}{1/\chi + (K-1)\ (1-\chi)} \frac{\tau_m}{\tau_0} \tag{4.70}$$

De observat că pentru $K = 1$, $\tau_m = \tau_0 / \chi$, astfel încât $t_p = t_s$.

Valoarea medie pentru τ_0 este cunoscută, dar determinarea valorii τ_m nu este simplă şi depinde de problemă. Dacă τ_0 este constant şi dacă $1/\chi$ este mult mai mare decât K, valoarea lui τ_m poate fi aproximată prin $(\tau_0 + \tau_c) / K\chi$, unde τ_c este timpul necesar unui procesor “slave” pentru comunicarea rezultatului.

Roussel-Ragot şi Dreyfus au utilizat algoritmul paralel de călire simulată pentru o problemă simplă de plasare a unei reţele bidimensionale de b^2 circuite pe o suprafaţă pătrată. În configuraţia iniţială a sistemului, fiecare circuit este conectat cu cei mai apropiaţi vecini prin conexiuni cu două terminale. Mutarea elementară este interschimbarea a două circuite, alese în mod arbitrar. Funcţia de cost este lungimea totală a conexiunilor. Parametrii utilizaţi sunt următorii:

- Configuraţia iniţială este aleasă în mod aleator.
- Temperatura iniţială este aleasă astfel încât rata de acceptare este de peste 0.9.
- Temperatura este modificată atunci când au fost evaluate $5 \cdot b^2 \cdot (b^2 - 1)$ mutări sau atunci când au fost acceptate $b^2 \cdot (b^2 - 1) / 2$ mutări.
- Parametrul de răcire α este egal cu 0.9.
- Procesul de călire simulată se opreşte atunci când temperatura atinge valoarea 0.2 sau când nu mai este acceptată nici o mutare la o temperatură dată.

Algoritmul a fost implementat pe o reţea de transputere. Un transputer a fost utilizat ca procesor “master”, calculele fiind efectuate pe transputerele conectate cu acest procesor.

4.5.2 Plasarea prin rețele neuronale artificiale

Interesul recent manifestat pentru reøelele neuronale are la bazã recunoašterea faptului cã creierul uman executã calcule într-un mod diferit de cel al calculatoarelor digitale. Calculatoarele pot fi extrem de rapide și precise în executarea secvenøelor de instrucøiuni care au fost formulate în mod explicit. Un sistem uman de procesare a informaøiilor este format din neuroni cu o vitezã de comutare de aproximativ un milion de ori mai redusã decât cea a porøilor logice. Cu toate acestea, un sistem uman este mult mai eficient decât calculatoarele în rezolvarea unor probleme complexe din punct de vedere computaøional, ca de exemplu înøelegerea vorbirii. De asemenea, omul poate procesa informaøii vizuale într-un mod mai eficient decât cele mai rapide calculatoare.

Reøelele neuronale artificiale au fost utilizate pentru rezolvarea unor probleme dificile, inclusiv pentru unele probleme de optimizare. În secøiunea 4.5.2.1 se prezintã unele concepte fundamentale și modele ale reøelelor neuronale artificiale. În secøiunea 4.5.2.2 se prezintã reøelele neuronale Hopfield care se pot utiliza pentru rezolvarea diferitelor probleme de optimizare, inclusiv a celei de plasare. În secøiunea 4.5.2.3 se prezintã modul în care se pot utiliza reøelele neuronale artificiale pentru rezolvarea problemei de plasare.

4.5.2.1 Concepte de bază ale rețelelor neuronale artificiale

O reøea neuronalã artificialã poate fi definitã ca o reøea sinteticã care emuleazã reøelele neuronale biologice ale organismelor vii. O altã definiøie este cã reøelele neuronale artificiale reprezintã o clasã de algoritmi matematici, deoarece o reøea poate fi consideratã ca o notaøie graficã pentru o clasã largã de algoritmi [190].

O reøea neuronalã artificialã este un ansamblu al unui mare numãr de *neuroni artificiali*. Prima definiøie formalã a unui model de neuron artificial bazatã pe o considerare foarte simplificatã a unui neuron biologic a fost formulatã de McCulloch și Pitts. Acest model este ilustrat în Figura 4.15.

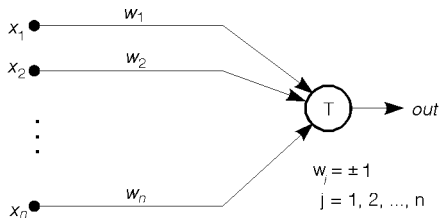

Figura 4.15. Modelul McCulloch-Pitts al neuronului.

Intrãrile x_i, pentru i = 1, 2, …, n, sunt 0 sau 1, depinzând de absenøa sau prezenøa impulsului de intrare la momentul de timp k. Semnalul de ieșire al neuronului este *out*. Regula de activare a acestui model este definitã astfel:

$$out^{k+1} = \begin{cases} 1 \text{ dacă } \sum_{i=1}^{n} w_i x_i^k \geq T \\ 0 \text{ dacă } \sum_{i=1}^{n} w_i x_i^k < T \end{cases} \tag{4.71}$$

unde indicii superiori k = 0, 1, 2, … indicã momentul discret de timp, iar w_i este ponderea conexiunii intrãrii i cu membrana neuronului. T este valoarea de prag a neuro-

nului, care trebuie depăşită de suma ponderată a intrărilor pentru ca neuronul să fie activat. Deşi acest model al neuronului este foarte simplist, are totuşi un important potenţial computaţional. Modelul poate executa operaţiile logice de bază NOT, OR şi AND, dacă valorile ponderilor şi ale pragurilor sunt selectate în mod corespunzător.

Modelul McCullogh-Pitts al unui neuron este caracterizat prin formalism şi o definiţie matematică precisă. Modelul utilizează însă 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 intrărilor şi valorile de prag ale neuronilor sunt fixe. Totuşi, acest model se poate utiliza pentru introducerea unor concepte de bază.

Un neuron artificial constă dintr-un *element de procesare* (*nod*), care recepţionează un număr de intrări analogice având conexiuni sinaptice, şi generează o singură ieşire. Simbolul general al unui neuron este indicat în Figura 4.16. Semnalele de intrare x_i ale neuronului sunt considerate unidirecţionale, ca şi semnalul de ieşire *out*. Fiecărei intrări x_i i se asociază o pondere w_i.

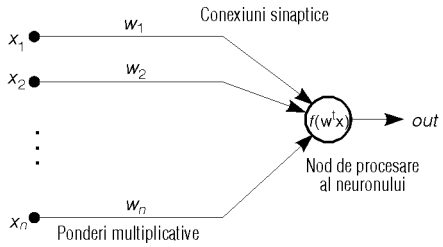

Figura 4.16. Simbolul general al neuronului constând din nodul de prelucrare şi conexiunile sinaptice.

Semnalul de ieşire al neuronului este dat de relaţia următoare:

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

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

unde **w** este vectorul ponderilor definit ca

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

iar **x** este vectorul de intrare:

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

Ambii vectori sunt vectori coloană. Funcţia $f(\mathbf{w}^T\mathbf{x})$ este numită *funcţie de activare* a neuronului. Domeniul acestei funcţii este setul valorilor de activare, *net*, ale modelului neuronului, de aceea această funcţie este utilizată adesea ca $f(net)$. Variabila *net* este definită ca un produs scalar al vectorului ponderilor şi al vectorului de intrare:

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

Argumentul funcţiei de activare, variabila *net*, este analogul potenţialului membranei neuronului biologic. De observat că valoarea de prag *T* nu este utilizată în mod explicit în ecuaţiile (4.72), (4.73) şi (4.74). Deoarece în unele modele valoarea de prag are un rol important, în aceste cazuri este necesară extragerea explicită a acestei valori ca un parametru separat al modelului neuronului. Până acum s-a presupus că neuronul modelat are n-1 conexiuni sinaptice de la variabilele de intrare x_1, x_2, …, x_{n-1}. S-a presupus de asemenea că $x_n = -1$ şi $w_n = T$.

Diferitele clase de reøele neuronale artificiale utilizeazã diferite funcøii *f* (*net*). De asemenea, chiar în cazul aceleiaši 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*).

Se observã din ecuaøiile (4.72) ši (4.73) cã neuronul, ca un nod de procesare, executã produsul scalar sau operaøia de însumare a intrãrilor sale ponderate pentru a obøine variabila *net*. Ulterior, executã operaøia neliniarã *f* (*net*) prin funcøia sa de activare. Funcøii de activare tipice sunt [190]

$$f(net) = \frac{2}{1 + e^{-\lambda\, net}} - 1 \tag{4.75}$$

ši

$$f(net) = \mathrm{sgn}(net) = \begin{cases} +1, & net > 0 \\ -1, & net < 0 \end{cases} \tag{4.76}$$

unde $\lambda > 0$ din ecuaøia (4.75) determinã panta funcøiei continue *f* (*net*) în apropiere de *net* = 0. Pentru $\lambda \to \infty$, limita funcøiei continue devine funcøia sgn (*net*) definitã în ecuaøia (4.76). Funcøiile de activare (4.75) ši (4.76) reprezintã funcøia *bipolarã continuã*, respectiv *bipolarã binarã*. Cuvântul “bipolar” indicã faptul cã se generează atât rãspunsuri pozitive, cât ši rãspunsuri negative ale neuronilor pentru aceste definiøii ale funcøiei de activare.

Prin deplasarea ši scalarea funcøiilor de activare definite de ecuaøiile (4.75) ši (4.76) se poate obøine funcøia de activare *unipolarã continuã*, respectiv *unipolarã binarã*, definite prin:

$$f(net) = \frac{1}{1 + e^{-\lambda\, net}} \tag{4.77}$$

ši

$$f(net) = \begin{cases} 1, & net > 0 \\ 0, & net < 0 \end{cases} \tag{4.78}$$

Funcøiile de activare (4.75) ši (4.77) se numesc adesea *caracteristici sigmoidale*, spre deosebire de funcøiile de activare (4.76) ši (4.78), care descriu modelul discret al neuronului.

Dacã funcøia de activare a neuronului are forma bipolarã binarã din ecuaøia (4.76), simbolul din Figura 4.16 se poate înlocui prin diagrama din Figura 4.17(a), care este o schemã bloc funcøionalã a unui neuron discret, indicând însumarea efectuatã de nodul de sumare ši limitarea efectuatã de unitatea logicã de prag.

În cazul funcøiei de activare continue cu forma din ecuaøia (4.75), modelul utilizat este ilustrat în Figura 4.17(b). Neuronul este reprezentat ca un amplificator sumator care amplificã semnalul de intrare $\mathbf{w}^T\mathbf{x}$. Modelele din Figura 4.17(a) ši (b) se numesc *perceptron discret* (*binar*), respectiv *perceptron continuu*. Perceptronul discret, introdus de Rosenblatt, a fost prima mašinã de învãøare, fiind precursorul multor modele neuronale actuale.

Pe baza definiøiei neuronului artificial, o reøea neuronalã artificialã poate fi definitã ca o interconexiune de neuroni astfel încât ieširile neuronilor sunt conectate, prin ponderi, cu ceilaløi neuroni; sunt permise conexiuni fãrã întârzieri sau cu întârzieri. Existã douã modele principale ale reøelelor neuronale artificiale: reøele directe (*feedforward*) ši reøele cu reacøie inversã (*feedback*).

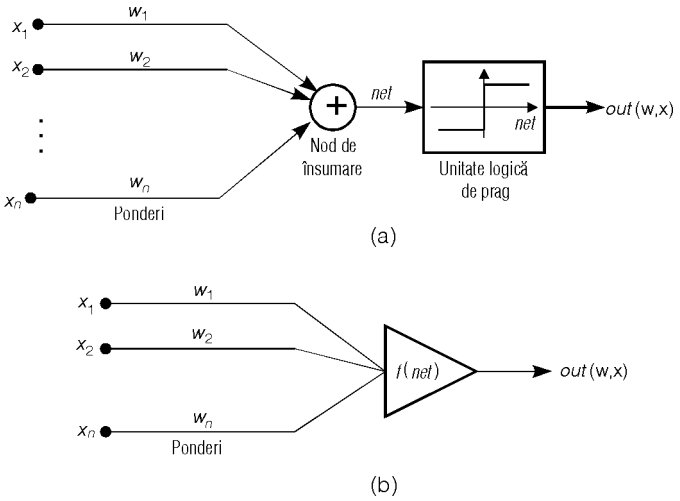

Figura 4.17. Modele ale neuronilor cu conexiuni sinaptice: (a) perceptron discret (binar); (b) perceptron continuu.

O reţea neuronalã directã formatã din *m* neuroni, fiecare având *n* intrări, este prezentatã în Figura 4.18. Vectorii de intrare şi de ieşire ai reţelei sunt:

$$\mathbf{x} = [x_1 \;\; x_2 \;\; \ldots \;\; x_n]^{\mathrm{T}} \tag{4.79}$$

$$\mathbf{out} = [out_1 \;\; out_2 \;\; \ldots \;\; out_m]^{\mathrm{T}} \tag{4.80}$$

Conexiunea cu ponderea w_{ij} conectează neuronul *i* cu intrarea *j*. Valoarea de activare pentru neuronul *i* se poate scrie ca:

$$net_i = \sum_{j=1}^{n} w_{ij} x_j, \text{ pentru } i = 1, 2, \ldots, m \tag{4.81}$$

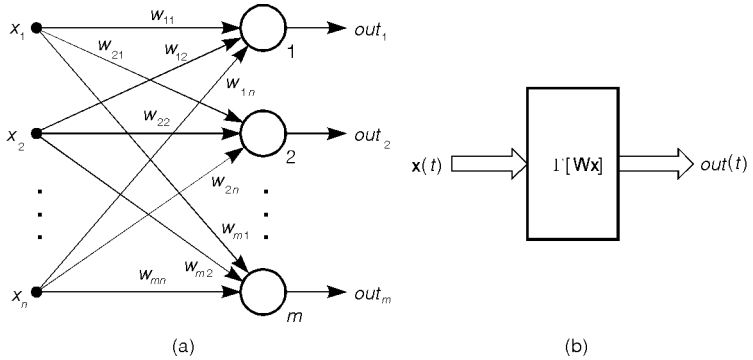

Figura 4.18. Reţea *feedforward* cu un singur strat: (a) schema de interconectare; (b) schema-bloc.

Următoarea transformare neliniarã este efectuatã de fiecare din cei *m* neuroni din reţea asupra vectorului de intrare **x**.:

$$out_i = f(\mathbf{w}_i^T \mathbf{x}), \text{ pentru } i = 1, 2, \ldots, m \tag{4.82}$$

unde vectorul ponderilor $\mathbf{w}_i$ conține ponderile conexiunilor care conduc spre nodul de ieșire *i*, și este definit astfel:

$$\mathbf{w}_i = [w_{i1} \;\; w_{i2} \;\; \ldots \;\; w_{in}]^{\mathrm{T}} \tag{4.83}$$

Dacă se introduce operatorul matricial neliniar Γ, maparea spațiului intrărilor **x** în spațiul ieșirilor **out** implementată de rețea se poate exprima astfel:

$$\mathbf{out} = \Gamma[\mathbf{W}\,\mathbf{x}] \tag{4.84}$$

unde **W** este *matricea ponderilor*, numită de asemenea *matricea de conexiuni*:

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} & \ldots & w_{1n} \\ w_{21} & w_{22} & \ldots & w_{2n} \\ \vdots & \vdots & \ldots & \vdots \\ w_{m1} & w_{m2} & \ldots & w_{mn} \end{bmatrix} \tag{4.85}$$

iar

$$\Gamma[\cdot] = \begin{bmatrix} f(\cdot) & 0 & \ldots & 0 \\ 0 & f(\cdot) & \ldots & 0 \\ \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & \ldots & f(\cdot) \end{bmatrix} \tag{4.86}$$

Funcțiile neliniare de activare $f(\cdot)$ de pe diagonala operatorului matricial Γ operează asupra valorilor de activare *net* ale fiecărui neuron. Fiecare valoare de activare este un produs scalar a unei intrări cu vectorul ponderilor.

Rețelele neuronale directe sunt caracterizate prin lipsa reacției inverse. Aceste rețele pot fi conectate în cascadă pentru a crea o rețea multistrat. Într-o asemenea rețea, ieșirea unui strat reprezintă intrarea pentru stratul următor.

O rețea neuronală cu reacție inversă poate fi obținută din rețeaua directă din Figura 4.18 prin conectarea ieșirilor neuronilor la intrările lor. Rezultatul este ilustrat în Figura 4.19.

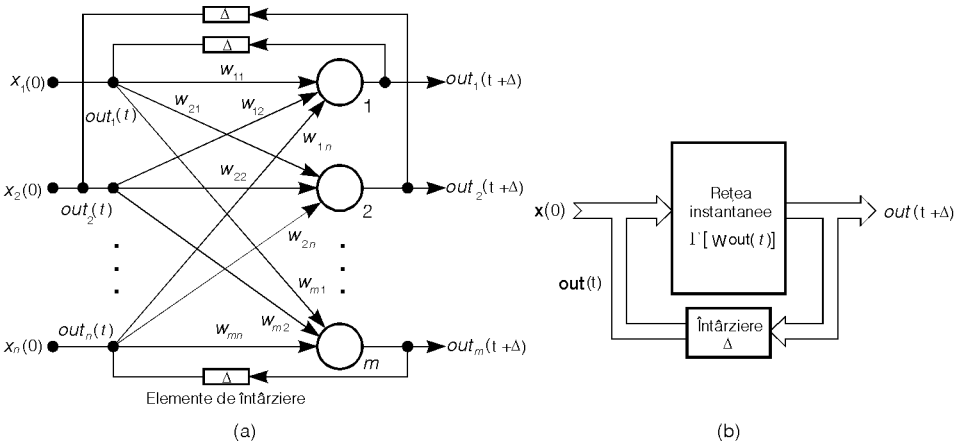

Figura 4.19. Rețea cu reacție inversă cu un singur strat: (a) schema de interconectare; (b) schema-bloc.

Rolul închiderii buclei de reacție este de a se permite controlul ieșirii out_i prin ieșirile out_j, pentru j = 1, 2, …, m. Un asemenea control este util dacă ieșirea prezentă, de exemplu **out**(t), controlează ieșirea de la următorul moment de timp, **out**(t+Δ).

Timpul Δ dintre momentele t şi $t + \Delta$ este introdus prin elementele de întârziere din bucla de reacţie. Maparea ieşirii **out**(t) în **out**(t+Δ) se poate scrie ca:

$$\mathbf{out}(t+\Delta) = \Gamma[\mathbf{W} \cdot \mathbf{out}(t)] \tag{4.87}$$

Intrarea **x**(t) este necesară doar pentru a iniţializa reţeaua astfel încât **out**(0) = **x**(0). Intrarea este apoi eliminată şi sistemul rămâne autonom pentru $t > 0$. Aici se consideră deci un caz special al acestei configuraţii cu reacţie, astfel încât **x**(t) = **x**(0), şi nu este furnizată nici o intrare pentru $t > 0$.

Există două categorii principale de reţele cu reacţie inversă. Dacă se consideră timpul ca o variabilă discretă, performanţele reţelei fiind observate la momente de timp discrete Δ, 2Δ, 3Δ, …, sistemul este numit *cu timp discret*. Pasul de timp este considerat unitar, momentele de timp fiind indexate prin întregi pozitivi. Simbolul Δ are atunci semnificaţia întârzierii unitare. Pentru o reţea neuronală cu timp discret, ecuaţia (4.87) se transformă în

$$\mathbf{out}^{k+1} = \Gamma[\mathbf{W} \cdot \mathbf{out}^{k}], \text{ pentru } k = 1, 2, \ldots \tag{4.88}$$

unde k este momentul de timp.

Reţeaua din Figura 4.19 este numită *recurentă* deoarece răspunsul acesteia la momentul de timp k+1 depinde de întreaga istorie a reţelei începând de la $k = 0$. Reţelele recurente operează de obicei cu o reprezentare discretă a datelor. Un sistem cu intrări la momente discrete de timp şi cu o reprezentare discretă a datelor este numit un automat. Reţelele neuronale recurente din această clasă pot fi considerate deci ca automate [190].

Ecuaţia (4.88) descrie starea $\mathbf{out}^k$ a reţelei la momentele k = 1, 2, …, şi generează o secvenţă de tranziţii ale stărilor. Reţeaua începe tranziţiile stărilor după ce este iniţializată la momentul 0 cu $\mathbf{x}^0$, trecând printr-o succesiune de stări până când, eventual, ajunge într-o stare de echilibru. Această stare de echilibru este numită adesea *atractor*. Un atractor poate consta dintr-o singură stare sau un număr limitat de stări. Secvenţa de stări a unei reţele recurente este în general non-deterministică. În plus, pot exista mai multe stări de echilibru care pot fi atinse în mod potenţial de reţea după o secvenţă de asemenea tranziţii non-deterministice.

4.5.2.2 Reţele neuronale Hopfield

Reţelele neuronale Hopfield se pot utiliza pentru rezolvarea diferitelor probleme de optimizare, printre care şi problema de plasare, motiv pentru care sunt descrise pe scurt în această secţiune. Pentru asemenea probleme, reţelele neuronale de tip gradient sunt cele mai adecvate. La aceste reţele, funcţia de energie descreşte în timp, timpul fiind presupus o variabilă continuă.

O reţea neuronală Hopfield este caracterizată ca o reţea puternic interconectată de procesoare simple analogice. O asemenea reţea de tip gradient converge spre unul din minimele stabile din spaţiul stărilor [114] [190]. Evoluţia sistemului este în direcţia generală a gradientului negativ a unei funcţii de energie. În mod tipic, funcţia de energie a reţelei este echivalată cu o anumită funcţie obiectiv (penalizare) care trebuie minimizată. Căutarea unei energii minime executată de o reţea de tip gradient corespunde căutării unei soluţii a unei probleme de optimizare.

Pentru rezolvarea unor probleme de tipul celei a comis-voiajorului, Hopfield şi Tank au definit o funcţie de energie cuadratică:

$$E(\mathbf{V}) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} T_{ij}V_iV_j - \sum_{i=1}^{N} V_iI_i \tag{4.89}$$

unde $\mathbf{V} = (V_1, \ldots, V_N)$ reprezintă starea reţelei neuronale. V_i este ieşirea neuronului i. T_{ij} este intensitatea conexiunii sinaptice între neuronul V_i şi V_j, cu $T_{ii} = 0$. I_i reprezintă

curentul de intrare. Reøeaua neuronalã necesitã conexiuni simetrice pentru a asigura convergenøa numai în stãrile de echilibru.

Pentru rezolvarea problemelor de programare liniarã, Tank øi Hopfield au definit o funcøie de energie liniarã:

$$E(\mathbf{V}) = \mathbf{A}^T\mathbf{V} + \sum_{j=1}^{m} F(\mathbf{W}_j^T\mathbf{V} - b_j) \tag{4.90}$$

unde $\mathbf{A}^T\mathbf{V}$ este funcøia de cost supusã condiøiei restrictive $\mathbf{W}_j^T\mathbf{V} \geq b_j$. *F* este o funcøie neliniarã a cãrei derivatã *f* este definitã ca

$$f(x) = \begin{cases} 0, & x \geq 0 \\ x, & x < 0 \end{cases}$$

Fiecare minim stabil este un atractor în spaøiul stãrilor. Setul stãrilor iniøiale care iniøiazã evoluøia terminatã într-un atractor este numit *rezervor de atracøie*. Atractorul øi rezervorul fiecãrui atractor sunt determinate de funcøia de energie øi de parametrii interni ai reøelei neuronale. O reøea neuronalã Hopfield reprezintã o procedurã de cãutare localã deterministicã. Dupã selectarea stãrii iniøiale, reøeaua va converge la starea minimã în rezervorul cãreia se aflã starea iniøialã [114].

Ca un exemplu simplu de reøea neuronalã Hopfield, presupunem cã valoarea analogicã de intrare *x* trebuie convertitã în reprezentarea sa binarã printr-o reøea cu doi neuroni. Se presupune cã funcøia de activare este continuã unipolarã, fiind definitã între 0 øi 1. Eroarea de conversie A/D E_c poate fi consideratã ca o funcøie de energie, care poate fi exprimatã prin urmãtoarea relaøie:

$$E_c = \frac{1}{2}\left(x - \sum_{i=0}^{1} V_i 2^i \right)^2 \tag{4.91}$$

unde $V_0 + 2V_i$ este valoarea zecimalã a vectorului binar $[V_1 \;\; V_0]^T$, øi corespunde valorii analogice *x*. Motivul acestei alegeri a funcøiei de energie este cã minimizarea energiei va minimiza simultan øi eroarea de conversie. Scopul este determinarea matricii **W** øi a vectorului curenøilor de intrare **I** a unei reøele care minimizeazã eroarea E_c.

Evaluarea erorii din ecuaøia (4.91) indicã faptul cã aceasta conøine termenii V_0^2 øi V_1^2, ceea ce face ca termenii w_{ii} sã fie egali cu 2^{2i} în loc de zero. De aceea, se va adãuga un termen suplimentar de eroare E_a. În plus faøã de eliminarea elementelor diagonale ale matricii **W**, acest termen va fi minimizat în apropierea coløurilor pãtratului [0, 1] din planul V_0, V_1. De aceea, o alegere potrivitã pentru E_a este:

$$E_a = -\frac{1}{2}\sum_{i=0}^{1} 2^{2i} V_i (V_i - 1) \tag{4.92}$$

Prin combinarea funcøiilor de eroare E_a øi E_c rezultã urmãtoarea funcøie totalã de energie, care este minimizatã de convertorul A/D [190]:

$$E = \frac{1}{2}\left(x - \sum_{i=0}^{1} V_i 2^i \right)^2 - \frac{1}{2}\sum_{i=0}^{1} 2^{2i} V_i (V_i - 1) \tag{4.93}$$

Primul termen reprezintã funcøia de eroare a intrãrii *x* øi a ieøirii binare (V_0, V_1). Al doilea termen este funcøia de restricøie care forøeazã ca ieøirea sã fie 0 sau 1. Aceastã expresie a energiei trebuie egalatã cu formula generalã a energiei, care este egalã în acest caz cu:

$$E = -\frac{1}{2}\sum_{i=0}^{1}\sum_{j\neq i,j=0}^{1} w_{ij}V_iV_j - \sum_{i=0}^{1} I_iV_i \tag{4.94}$$

Prin rearanjarea membrului drept al expresiei energiei din ecuaţia (4.94) se obţine:

$$E = \frac{1}{2}x^2 + \frac{1}{2}\sum_{i=0}^{1}\sum_{j\neq i,j=0}^{1} 2^{i+j}V_iV_j + \sum_{i=0}^{1}\left(2^{2i-1} - 2^i x\right)V_i \tag{4.95}$$

Atunci când intrarea analogicã x este fixatã, termenul (½)x^2 este o constantã aditivã în ecuaţia (4.95), şi nu este relevantã pentru minimizarea funcţiei E indicate de ecuaţia (4.95) în planul V_0, V_1; de aceea, acest termen va fi omis. Comparând coeficienţii a2semãnãtori ai V_0 şi V_1 din partea dreaptã a funcţiilor de energie (4.94) şi (4.95), rezultã urmãtorii parametri ai reţelei:

$$\begin{aligned} w_{01} &= w_{10} = -2 \\ I_0 &= x - \frac{1}{2} \\ I_1 &= 2x - 2 \end{aligned} \tag{4.96}$$

Reţelele neuronale de tip gradient sunt exemple de sisteme neliniare, dinamice, şi asimptotic stabile. Aceste reţele neuronale nu garanteazã însã convergenţa la minimul global, din cauza faptului cã nu dispun de posibilitatea unei cãutãri globale. Pentru a se evita blocarea în minime locale, unii autori au propus utilizarea procedurii de cãlire simulatã pentru reţelele neuronale. Deoarece aceastã procedurã este secvenţialã, ea necesitã un timp de calcul semnificativ pentru obţinerea unei soluţii optime. O altã problemã a acestor reţele este cã un mare numãr de parametri trebuie selectaţi şi ajustaţi în mod corespunzãtor chiar şi pentru a se asigura convergenţa la o soluţie validã.

Avantajul reţelelor neuronale Hopfield este convergenţa rapidã la un minim stabil. Dacã la funcţia de energie se adaugã condiţii de restricţie cu o pondere mare, punctul iniţial va converge spre cel mai apropiat punct valid fãrã influenţã din partea funcţiei de cost. Aceasta diferã de cazul algoritmilor genetici, deoarece acешti algoritmi nu pot ajusta configuraţia spre o soluţie validã, chiar dacã penalizarea este severã. Aceastã proprietate face ca reţelele neuronale Hopfield sã poatã fi utilizate pentru cãutarea localã în cadrul algoritmilor genetici pentru rezolvarea problemelor combinaţionale şi a problemelor cu restricţii. Pe aceastã bazã, Lin *et al.* [114] au propus un model hibrid care utilizeazã algoritmi genetici şi reţele neuronale Hopfield pentru optimizarea funcţiilor.

4.5.2.3 Utilizarea reţelelor neuronale pentru plasare

Considerãm cazul cel mai simplu al problemei de plasare. Fiind date n module de circuit şi o matrice de conectivitate **C** = [C_{ij}], unde C_{ij} indicã conectivitatea între modulul i şi modulul j, cele n module trebuie plasate în n locaţii ale unei suprafeţe bidimensionale, astfel încât lungimea totalã de interconectare Manhattan sã fie minimizatã. Metoda se va ilustra prin circuitul din Figura 4.20(a), locaţiile în care trebuie plasate modulele fiind ilustrate în Figura 4.20(b).

Soluţia acestei probleme aparţine lui Yu [187], care a utilizat reţelele neuronale Hopfield pentru rezolvarea problemei de plasare. Acesta a modificat soluţia propusã de Hopfield şi Tank pentru rezolvarea problemei comis-voiajorului. Se utilizeazã o reţea cu n^2 neuroni. Reţeaua constã dintr-o matrice de $n \times n$ neuroni, dupã cum se prezintã în Figura 4.21.

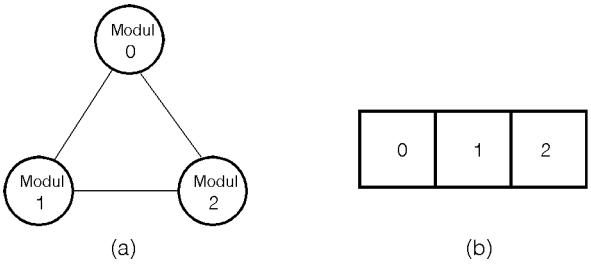

Figura 4.20. (a) Exemplu de graf al unui circuit. (b) Definiøia poziøiilor.

Neuronii sunt numerotaøi de la 0 la n^2 - 1, de la stânga la dreapta şi de sus în jos. Valoarea unui element din poziøia (i, j) a matricii reprezintã "şansa" 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 ieşirea 1. Ieşirile neuronilor sunt normalizate, astfel încât acestea au valori între 0 şi 1.

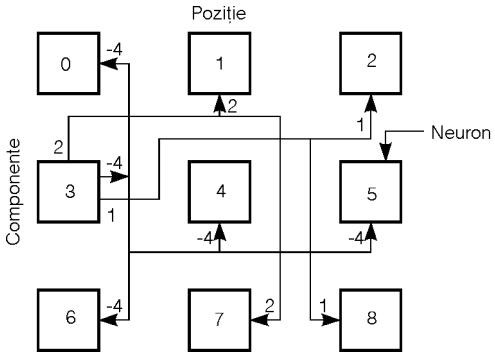

Figura 4.21. O reøea neuronalã artificialã utilizatã pentru plasare.

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 şi l (care este un element al matricii intensitãøii sinapselor) este definitã ca şi conectivitatea între modulele de circuit i_1 şi i_2 multiplicatã cu f (j_1, j_2), unde f este o funcøie a distanøei între locaøiile j_1 şi 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 şi j_2), unde parametrul *offset* este de obicei mai mare decât $\sqrt{n}$.

De exemplu, intensitatea sinapsei dintre neuronii 2 şi 3 ($T_{2,3}$) se poate determina astfel. Neuronul 2 are (i_1, j_1) = 0, 2, iar neuronul 3 are (i_2, j_2) = 1, 0 (Figura 4.26). Astfel, $T_{2,3}$ va fi egal cu

$C_{0,1} \times$ (*offset* - distanøa Manhattan dintre 2 şi 0) = $1 \times (3 - 2) = 1$

unde s-a presupus *offset* = 3.

Conexiunile corespunzãtoare ale neuronului 3 sunt prezentate în Figura 4.21.

Determinarea parametrilor reţelei neuronale (matricea ponderilor, valorile pragurilor, constantele implicate în funcţia de energie şi funcţia de activare) este dificilă, soluţia finală depinzând în mare măsură de acești parametri. De acești parametri depinde de asemenea convergenţa reţelei spre o soluţie validă.

Rezultatele obţinute de Yu în utilizarea reţelelor neuronale Hopfield pentru rezolvarea problemei de plasare nu au fost promiţătoare. Unele din dificultăţile indicate au fost timpii mari de simulare, calitatea redusă a soluţiilor și sensibilitatea soluţiilor de parametrii reţelei. Pe de altă parte, Sriram și Kang au obţinut rezultate încurajatoare la plasarea bidimensională a unui număr de până la 64 de module, reţeaua utilizată fiind capabilă să determine soluţiile optime sau apropiate de cele optime în cele mai multe cazuri [190].

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

O plasare adecvată pentru circuitele FPGA cu resurse limitate de rutare trebuie să realizeze nu numai minimizarea lungimii conexiunilor prin gruparea celulelor care sunt interconectate, dar trebuie să asigure și rutabilitatea circuitelor. În această secţiune se prezintă algoritmi de plasare elaboraţi pentru asemenea circuite FPGA, care au ca obiectiv asigurarea rutabilităţii circuitelor. În secţiunea 4.6.1 se prezintă un algoritm de plasare bazat pe metoda de partiţionare prin tăietura minimă, care încearcă asigurarea rutabilităţii prin distribuirea uniformă a conexiunilor în cadrul partiţiilor. În secţiunea 4.6.2 se prezintă un algoritm genetic pentru plasare, care alocă un număr de celule libere în procesul de plasare, care vor fi utilizate pentru rutare.

4.6.1 Algoritm de plasare pe baza tăieturii minime

4.6.1.1 Descrierea algoritmului de plasare

Tehnicile de plasare cele mai utilizate se bazează pe partiţionarea prin metoda tăieturii minime. Acestea aplică în mod recursiv o procedură de bipartiţionare cu scopul minimizării numărului de interconexiuni care intersectează linia de tăietură în fiecare etapă a algoritmului. Algoritmul realizează optimizarea prin plasarea apropiată a celulelor care sunt interconectate. De aceea, componenta principală a funcţiei de cost pentru un asemenea algoritm este lungimea interconexiunilor.

Se prezintă în continuare un algoritm de plasare bazat pe metoda de partiţionare descrisă în capitolul 3 (secţiunea 3.7.1), cu o funcţie obiectiv care urmărește pe lângă reducerea lungimii interconexiunilor, și distribuirea uniformă a conexiunilor în cadrul partiţiilor. Ca urmare, rezultatul plasării maximizează posibilitatea unei rutări fezabile a circuitului. Este studiat de asemenea efectul utilizării diferitelor secvenţe de aplicare a liniilor de tăietură, cu scopul reducerii numărului de conexiuni în apropierea centrului circuitului, deoarece în cazul plasării obișnuite care se bazează pe metoda tăieturii minime apare în mod frecvent creșterea numărului de conexiuni în această zonă.

Algoritmii de plasare care utilizează partiţionarea ierarhică pe baza tăieturii minime au fost utilizaţi în mod frecvent. Fiind dat un circuit reprezentat printr-un graf, fiecare vârf reprezentând o celulă logică și fiecare muchie reprezentând o conexiune, acești algoritmi partiţionează în mod recursiv circuitul și suprafaţa de plasare, până când se ajunge la un graf cu un singur nod, care va fi plasat într-o regiune a suprafeţei. Obiectivul fiecărei etape de bipartiţionare este minimizarea dimensiunii tăieturii, cu restricţia ca fiecare porţiune să conţină același număr de noduri, sau un număr

apropiat. Aceastã metodã are avantajul obøinerii unei plasãri de calitate corespunzãtoare, a unui timp de execuøie redus şi a unei implementãri simple.

Dezavantajul algoritmilor de plasare amintiøi este cã dimensiunea tãieturii este singura metricã utilizatã în cadrul funcøiei de cost. De aceea, este posibilã obøinerea unei dimensiuni reduse a tãieturii, şi în acelaşi timp a unor porøiuni cu un numãr de conexiuni semnificativ diferit. Ca urmare, o asemenea plasare poate fi rutatã într-un mod dificil, sau efectuarea rutării poate fi chiar imposibilã. Pentru a elimina acest dezavantaj, este necesar ca în cadrul funcøiei de cost sã se ia în considerare nu numai dimensiunea tãieturii, ci şi distribuøia interconexiunilor din cele douã porøiuni ale bipartiøiei.

Algoritmul de plasare propus utilizeazã bipartiøionarea a cãrei funcøie obiectiv urmãreşte minimizarea lungimii interconexiunilor simultan cu minimizarea diferenøei între numãrul de conexiuni din cele douã porøiuni. Aceastã diferenøã, numitã numãr de dezechilibru, este mãsura care urmãreşte distribuøia echilibratã a conexiunilor din cele douã porøiuni. În cadrul algoritmului de plasare se aplicã în mod recursiv bipartiøionarea cu funcøia de cost descrisã în secøiunea 3.7.1, pânã când se ajunge la porøiuni care conøin o singurã celulã a circuitului. Liniile de tãieturã se aplicã în mod alternativ, pe orizontalã şi pe verticalã.

4.6.1.2 Secvenøa de aplicare a liniilor de tãieturã

Pe lângã o procedurã eficientã de partiøionare, este necesarã o strategie adecvatã de aplicare a liniilor de tãieturã. O secvenøã tradiøionalã de aplicare a liniilor de tãieturã, cum este procedura de partiøionare cuadraticã [146], are dezavantajul cã nu øine cont de poziøia terminalelor externe. Pentru aceasta se poate utiliza tehnica numitã propagarea terminalelor. Aceastã tehnicã are însã dezavantaje. De exemplu, cele douã regiuni sunt procesate secvenøial, neexistând criterii pentru stabilirea regiunii care trebuie procesatã prima.

Secvenøa de aplicare a liniilor de tãieturã are un rol important. În cadrul plasării convenøionale bazatã pe tãietura minimã, se alege linia de tãieturã poziøionatã în centrul regiunii curente. Dacã liniile de tãieturã se aplicã în aceastã ordine, Figura 4.22 indicã secvenøa liniilor de tãieturã, unde liniile întrerupte sunt cele aplicate recent, iar liniile continue sunt cele aplicate anterior.

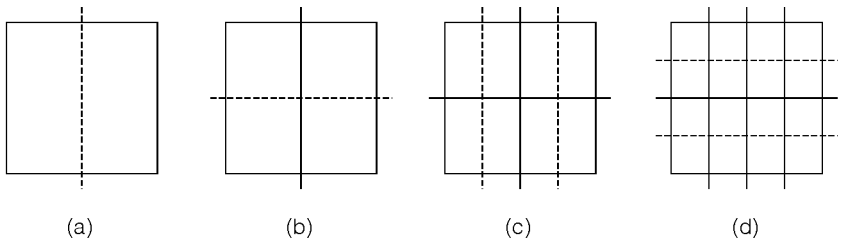

Figura 4.22. Alegerea liniei de tãieturã poziøionatã în centru.

Presupunem cã linia curentã de tãieturã, cea indicatã în Figura 4.23 printr-o linie întreruptã, este în imediata apropiere a liniei de tãieturã din centru. Aceastã linie va înjumãtãøi patru regiuni şi conexiunile corespunzãtoare. În fiecare etapã a acestei bipartiøionãri, se pot interschimba perechi de noduri, de exemplu astfel încât unul din noduri este în regiunea *A*, iar al doilea în *E*. 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.

Numărul de noduri dintr-o regiune este proporţional cu suprafaţa regiunii respective. Deoarece linia de tăietură considerată este în apropierea liniei de tăietură din centru, regiunile intersectate au o suprafaţă redusă. De aceea, numărul nodurilor din aceste regiuni este redus. În consecinţă, numărul perechilor posibile care pot fi interschimbate în procesul de bipartiţionare este limitat. De obicei, aceasta are ca rezultat o dimensiune a tăieturii relativ ridicată pentru liniile de tăietură din apropierea centrului, din cauza numărului mic de mutări posibile.

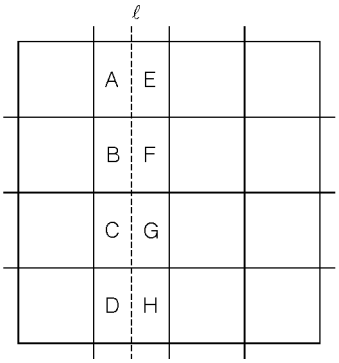

Figura 4.23. Plasare tradiţionalã pe baza tăieturii minime.

Din cele de sus rezultă că, pentru a se reduce dimensiune tăieturii în apropierea centrului, în această zonă liniile de tăietură trebuie aplicate în primele etape ale procesului de bipartiţionare, după cum se indică în Figura 4.24. Pentru claritate, liniile de tăietură orizontale nu sunt indicate în această figură. Liniile de tăietură orizontale şi verticale sunt aplicate alternativ şi în acest caz.

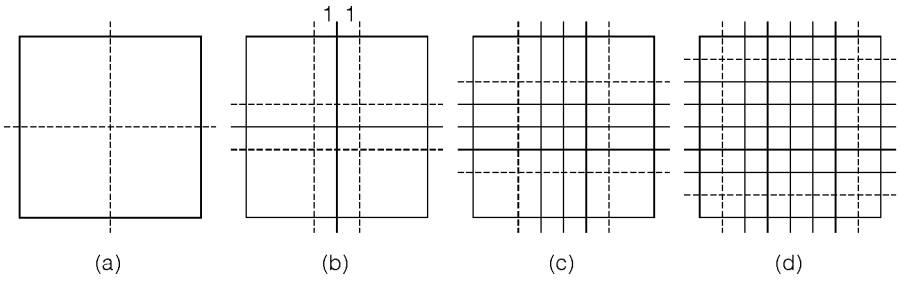

Figura 4.24. Secvenţa propusă pentru aplicarea liniilor de tăietură.

4.6.2 Algoritm genetic pentru plasarea circuitelor FPGA

Algoritmii genetici reprezintă o paradigmă eficientă pentru rezolvarea problemelor complexe de optimizare. După cum s-a arătat de J. H. Holland, simularea evoluţiei naturale în care populaţia trece printr-un proces de adaptare, cu o strategie controlată de evoluţie, poate fi o alternativă eficientă pentru rezolvarea acestui tip de probleme. În timp ce alţi algoritmi îmbunătăţesc în mod iterativ o plasare iniţială prin mutarea unei singure celule sau prin interschimbarea a două celule, un algoritm genetic porneşte de la un set de plasări iniţiale care reprezintă populaţia iniţială. Algoritmul încearcă să combine caracteristicile adecvate din două configuraţii 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 cunoøtinøele 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, cum sunt circuitele FPGA *Atmel* 6000 [7]. 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

Deøi nu au fost utilizaøi pentru plasarea circuitelor FPGA, algoritmii genetici au fost utilizaøi pentru plasarea circuitelor VLSI. Lucrarea clasicã a fost realizatã de Cohoon *et al.* [57] [58]. Aceøtia au codificat o plasare prin notaøia polonezã a unui arbore binar, un cromozom fiind deci reprezentat printr-un øir. Au fost utilizaøi diferiøi operatori de recombinare, care opereazã fie direct asupra øirurilor, 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 øi Mazumder [74] au raportat un algoritm numit SAGA, care este o generalizare a algoritmului genetic øi a algoritmului de cãlire simulatã. În funcøie de setarea parametrilor sãi de control, SAGA se comportã ca un algoritm genetic, un algoritm de cãlire simulatã, sau o combinaøie a acestora. Autorii au arãtat experimental cã prin mixarea algoritmului genetic cu algoritmul de cãlire simulatã se obøine o plasare de calitate mai bunã decât printr-un algoritm pur genetic.

Mohan øi Mazumder [124] au descris un algoritm genetic distribuit pentru plasarea celulelor standard. Algoritmul a fost elaborat pentru a fi executat pe o reøea de staøii de lucru. Procedura de plasare distribuitã executã un algoritm genetic de bazã pe fiecare procesor din reøea. Este introdus un nou operator genetic, *migraøia*, care transferã informaøii de plasare de la un procesor la altul în cadrul reøelei.

Schnecke øi Vornberger [150] au prezentat un algoritm genetic pentru proiectarea fizicã a circuitelor VLSI. Algoritmul combinã etapele de amplasare øi de rutare, optimizând simultan plasarea celulelor øi rutarea. Aceiaøi autori au descris în [149] un algoritm genetic paralel pentru optimizarea combinatã a plasãrii øi a rutãrii. Procesul de cãutare este auto-adaptabil. Se executã mai muløi algoritmi genetici secvenøiali, cu strategii diferite. La intervale fixe de timp, aceste strategii sunt evaluate øi fiecare strategie este ajustatã pentru a obøine rezultate mai bune. Ambii algoritmi [149], [150] utilizeazã un genotip codificat ca un arbore binar, care defineøte plasarea relativã a celulelor.

Considerăm graful din Figura 4.25(a) ale cărui vârfuri reprezintă module, iar numerele din dreptul muchiilor reprezintă ponderi ale interconexiunilor. Cele nouă module pot fi plasate în nouă locaţii, care sunt definite în Figura 4.25(b). O soluţie posibilă a problemei de plasare este indicată în Figura 4.25(c).

Soluţia din Figura 4.25(c) poate fi reprezentată ca un şir de simboluri. Dacă primul simbol din stânga şirului corespunde locaţiei 0 din Figura 4.25(b), iar simbolul din dreapta şirului corespunde locaţiei 8, soluţia din Figura 4.25(c) poate fi reprezentată prin şirul [AGHCBIDEF] $\left(\frac{1}{85}\right)$. Numărul din paranteză reprezintă valoarea funcţiei de viabilitate, care este inversa lungimii ponderate a conexiunilor pe baza metricii Manhattan [146].

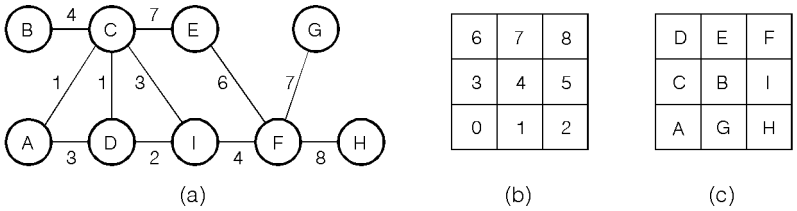

Figura 4.25. (a) Graful unui circuit. (b) Definiţia poziţiilor. (c) O plasare posibilă.

Dacă locaţia din colţul din stânga jos al Figurii 4.25(b) este considerată în originea sistemului de coordonate cartezian, se pot calcula în mod simplu coordonatele oricărui modul pe baza indexului în şirul care reprezintă soluţia plasării. De exemplu, indexul modulului E este 7. Coordonatele carteziene ale acestui modul sunt date de x = (7 mod 3) = 1, iar $y = \left\lceil \frac{7}{3} \right\rceil = 2$.

Orice şir de lungime 9 care conţine caracterele [A, B, C, D, E, F, G, H, I] reprezintă o soluţie posibilă. Alte soluţii posibile (cromozomi) sunt [BDEFIGCHA] $\left(\frac{1}{110}\right)$, [IHAGBFCED] $\left(\frac{1}{95}\right)$, [BIDEFAGHC] $\left(\frac{1}{86}\right)$.

Şi în cazul plasării, operatorii genetici cei mai utilizaţi sunt încrucişarea, mutaţia şi inversiunea.

Cel mai simplu operator de *încrucişare* constă în alegerea aleatoare a unui punct de tăietură şi generarea urmaşului prin combinarea segmentului din stânga punctului de tăietură al unui părinte cu segmentul din dreapta punctului de tăietură al celuilalt părinte. Din exemplul anterior, considerăm cei doi părinţi [BIDEF.AGHC] $\left(\frac{1}{86}\right)$ şi [BDEFI.GCHA] $\left(\frac{1}{110}\right)$. Dacă punctul de tăietură este ales după poziţia 4, urmaşul produs va fi [BIDEF.GCHA]. Lungimea ponderată a conexiunilor pentru urmaşul generat este redusă la 63, şi astfel funcţia de viabilitate a urmaşului are valoarea $\frac{1}{63}$.

În exemplul anterior, elementele de la stânga punctului de tăietură al unui părinte nu apar la dreapta punctului de tăietură al celuilalt părinte. În caz contrar, unele simboluri din şirul soluţiei ar fi repetate. În cazul plasării celulelor aceasta nu ar reprezenta o soluţie fezabilă. Există diferite modificări ale operaţiei de încrucişare descrise anterior în scopul evitării repetiţiei simbolurilor, de exemplu: (a) încrucişare ordonată, (b) încrucişare mapată parţial (*PMX*), şi (c) încrucişare ciclică.

Se exemplifică două operaţii de încrucişare utilizate în sistemul de plasare *Genie*, care a fost elaborat pentru plasarea modulelor într-o reţea rectangulară [57]. Primul operator de încrucişare selectează un modul aleator e_s din părintele 1 şi mută acest modul şi vecinii săi din acest părinte în locaţiile corespunzătoare vecinilor modulului din părintele 2. Modulele care ocupau locaţiile vecinilor din părintele 2 sunt deplasate apoi cu câte o locaţie în direcţia vechilor locaţii ale modulelor care au fost

mutate pânã când este gãsitã o locaøie liberã. Aceste deplasãri sunt ilustrate în Figura 4.26. Rezultatul este cã se copiazã din pãrintele 1 în pãrintele 2 o porøiune conøinând e_s øi cei patru vecini ai sãi, øi cã alte module sunt deplasate cu cel mult o poziøie.

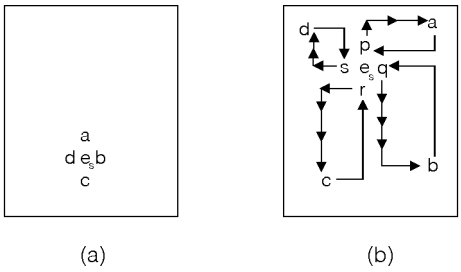

Figura 4.26. (a) Un modul aleator øi vecinii acestuia. (b) Vecinii din (a) ai pãrintelui 1 înlocuiesc modulele vecine ale pãrintelui 2.

Al doilea operator de încrucišare utilizat în sistemul *Genie* selecteazã un pãtrat conøinând $k \times k$ module din pãrintele 1 øi îl copiazã în pãrintele 2. Numãrul k este ales în mod aleator, având o medie de 3. Aceastã metodã are tendinøa sã dubleze unele module. De exemplu, în Figura 4.27, dacã modulele din pãtratul pãrintelui 1 sunt copiate în pãtratul pãrintelui 2, acele module din pãtratul pãrintelui 1 care nu se aflã în pãtratul pãrintelui 2 vor fi duplicate. Aceastã problemã este rezolvatã astfel: fie SP_2 - SP_1 setul modulelor din pãtratul pãrintelui 2 care nu se aflã în pãtratul pãrintelui 1 (SP_2 - SP_1 = {*x*, *w*, *p*, *m* }). Similar, SP_1 - SP_2 este setul modulelor din pãtratul pãrintelui 1 care nu se aflã în pãtratul pãrintelui 2 (SP_1 - SP_2 = {*c*, *d*, *e*, *f* }). Fiecare modul din SP_2 - SP_1 este mutat într-o locaøie ocupatã în acel moment de un modul din SP_1 - SP_2. Deci, în pãrintele 2 modulele {*x*, *w*, *p*, *m* } sunt mutate în locaøiile ocupate de modulele {*c*, *d*, *e*, *f* }. Toate modulele din pãtratul pãrintelui 1 sunt copiate apoi în pãtratul pãrintelui 2 pentru a se obøine un nou urmaš.

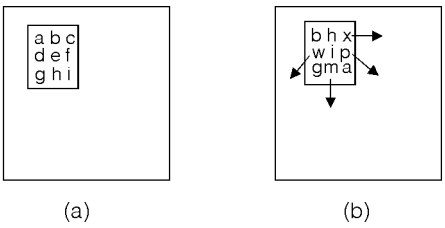

Figura 4.27. (a) Este selectat un pãtrat în pãrintele 1. (b) Modulele pãtratului din pãrintele 1 sunt copiate în pãrintele 2 øi modulele duplicate sunt eliminate.

Mutaøia produce modificãri incrementale aleatoare ale urmašului generat prin încrucišare. În cazul plasãrii, mecanismul de mutaøie cel mai utilizat constã în interschimbarea perechilor. Pentru plasare, o genã constând dintr-un triplet ordonat format dintr-o celulã øi coordonatele asociate acesteia nu poate fi prezentã în nici un alt individ al populaøiei. În acest caz, nu este utilã utilizarea doar a operaøiei de încrucišare, deoarece aceasta reprezintã un mecanism de moštenire. Operatorul de mutaøie genereazã noi tripleøi celulã-coordonate. Dacã noii tripleøi sunt corespunzãtori, configuraøiile care le conøin sunt reøinute. Mutaøia este controlatã de un parametru numit rata mutaøiei M_r. O ratã redusã a mutaøiei înseamnã cã infuzia de noi gene este foarte scãzutã. O ratã M_r ridicatã va determina ca urmašii sã piardã asemãnarea cu pãrinøii, ceea ce înseamnã cã algoritmul se comportã ca un proces fãrã memorie, pierzându-øi abilitatea de a învãøa din istoria cãutãrii [153].

În cazul operaţiei de *inversiune*, se aleg în mod aleator două puncte de tăietură în cadrul cromozomului, şi se inversează simbolurile subşirului din secţiunea tăiatã. De exemplu, şirul [BID.EFGCH.A] (tăiat după poziţiile cu indexul 2 şi 7) va deveni după inversiune [BID.HCGFE.A]. Operaţia este executată astfel încât nu modifică soluţia reprezentată de cromozom, ci modifică numai reprezentarea acestuia. Deci, simbolurile şirului trebuie să aibă o interpretare independentă de poziţia lor [83] [153].

Până acum s-au considerat operaţii asupra unor soluţii posibile complete. O soluţie parţială este reprezentată de o schemă. Pentru exemplul din Figura 4.25, şirul [∗I∗EFH∗G∗] indică o plasare parţială, celulele din poziţiile 0, 2, 6, 8 fiind nedefinite. Deoarece celula F este puternic conectată cu celulele I, E, G şi H, iar aceste celule sunt adiacente cu celula F în soluţia parţială amintită, această schemă reprezintă o plasare parţială de calitate. O asemenea plasare parţială va îmbunătăţi funcţia de viabilitate a unui individ care o moştenešte. Astfel, o schemă este o plasare topologică a celulelor corespunzătoare, indicând poziţiile lor relative şi nu locaţiile lor exacte.

4.6.2.2 Reprezentarea soluției

Reprezentarea soluţiei are un rol major în proiectarea unui algoritm genetic eficient. Principalele caracteristici ale unei soluţii sunt următoarele [41]:

- Este întotdeauna un şir de gene;
- Trebuie să permită o mare diversitate într-o populaţie de dimensiuni reduse;
- Trebuie să permită aplicarea simplă a operatorilor genetici;
- Trebuie să permită calcularea simplă a funcţiei obiectiv.

Pentru soluţia problemei de plasare s-a ales reprezentarea printr-un şir de înregistrări. Fiecare înregistrare reprezintă o celulă, conţinând identificatorul celulei şi coordonatele (*x*, *y*) ale acesteia.

Poziţia înregistrării corespunzătoare unei celule nu determină întotdeauna poziţia fizică a celulei în circuit. Totuşi, în anumite puncte din algoritm, poziţia celulei este recalculată pe baza ordonării înregistrărilor celulelor în şir, după cum urmează. Pornind de la prima linie de celule din circuit, celulele sunt listate în ordine de la stânga la dreapta. La sfârşitul liniei, se trece la linia următoare, iar celulele sunt listate în ordine inversă. Procesul este continuat până când sunt listate toate celulele, schimbându-se direcţia după fiecare linie. Fiind dat un şir de înregistrări, celulelor li se pot asigna poziţii în cadrul liniilor, prin inversarea procesului anterior.

4.6.2.3 Operatori genetici

Încrucişarea este principalul operator genetic. Dacă Π_a şi Π_b sunt doi indivizi părinte, reprezentând plasări valide, operaţia de încrucişare este definită ca $\Pi_a \times \Pi_b \rightarrow \Pi_u$, unde Π_u reprezintă urmaşul, fiind o altă plasare validă. Încrucişarea combină scheme din ambii părinţi, şi astfel urmaşul moştenešte unele din caracteristicile părinţilor.

Cea mai simplă formă de încrucişare, descrisă în secţiunea 4.6.2.1, nu poate fi aplicată în acest caz, deoarece poate crea şiruri care nu au un corespondent fizic. De exemplu, o încrucişare simplă între două şiruri ABCD şi BDCA poate produce două noi şiruri ABCA şi BDCD. Fiecare din acestea are două caractere care se repetă şi un caracter lipsă. Dacă fiecare caracter corespunde unei celule în cadrul problemei de plasare, atunci într-un şir valid fiecare caracter trebuie să apară o singură dată. În consecinţă, sunt necesari operatori de încrucişare mai complecşi, care păstrează corectitudinea plasării. În algoritmul descris, s-a utilizat un operator numit încrucişare mapată parţial (*PMX*).

Încrucişarea *PMX* se executã astfel: se selectează doi pãrinøi (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 există deja în cadrul urmaşului, este determinatã poziøia sa în pãrintele 2, şi este copiatã gena din poziøia determinatã a pãrintelui 1.

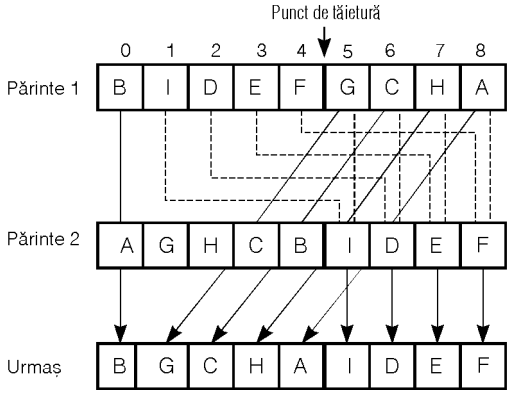

Figura 4.28. Încrucişarea *PMX*.

Figura 4.28 exemplificã modul în care se executã încrucişarea *PMX*. Fie punctul de tãieturã dupã poziøia 4. Subşirul din dreapta al pãrintelui 2, care este IDEF, este copiat în cadrul urmaşului. Apoi, primul pãrinte este parcurs de la stânga, şi deoarece gena B (poziøia 0) nu există în cadrul urmaşului, este copiatã în poziøia 0. Urmãtoarea genã, I (poziøia 1) există în poziøia 5. Gena din poziøia 5 a pãrintelui 1 este G, care nu există în cadrul urmaşului, şi de aceea gena G este copiatã în poziøia 1. Parcurgerea primului pãrinte se continuã în acelaşi mod, urmaşul generat fiind BGCHAIDEF.

Este esenøial ca indivizii cu viabilitate mai ridicatã sã participe la reproducere mai frecvent. În cazul în care un individ se reproduce de un numãr mare de ori în primele etape ale algoritmului genetic, este probabil cã toate soluøiile din populaøie vor moşteni anumite trãsãturi ale acestui individ şi astfel vor fi asemãnãtori. De aceea, strategia de selecøie a pãrinøilor trebuie sã evite această convergenøã prematurã a algoritmului genetic la un optim local. În cadrul algoritmului aceasta se obøine prin faptul cã unui pãrinte cu viabilitate ridicatã nu i se permite reproducerea decât o singurã datã într-o generaøie.

Mutaøia produce modificări aleatoare incrementale ale urmaşului generat prin încrucişare. Pentru problema de plasare, ca tehnicã de mutaøie se poate utiliza interschimbarea perechilor. Este posibil ca o genã constând dintr-un triplet ordonat al unei celule şi coordonatele ideale asociate acesteia sã nu fie prezentã în niciunul din indivizii populaøiei. În acest caz, încrucişarea nu este suficientã, deoarece este doar un mecanism de moştenire. Operatorul de mutaøie generează noi tripleøi celulã-coordonate. În cazul în care aceştia se comportã în mod corespunzãtor, configuraøiile care le conøin vor fi reøinute.

Mutaøia este controlatã de un parametru numit rata de mutaøie M_r. O valoare redusã a acestei rate înseamnã cã infuzia noilor gene este foarte redusã. Un asemenea nivel redus de mutaøie va modifica structura soluøiei numai într-o micã mãsurã, O valoare ridicatã a ratei de mutaøie va determina ca urmaşii sã-şi piardã asemãnarea cu pãrinøii acestora. 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 operaţie care modifică lungimea efectivă a unei scheme fără a altera viabilitatea individului în scopul creşterii probabilităţii de supravieţuire a schemelor mai lungi. Operatorul de inversiune modifică poziţiile înregistrărilor celulelor în şirul care reprezintă o plasare, dar nu modifică poziţia fizică a celulelor din circuit. Inversiunea este controlată de un parametru numit rata de inversiune I_r.

4.6.2.4 Selecţia populaţiei pentru generaţia următoare

Algoritmii genetici convenţionali utilizează metoda prin care se generează doi urmaşi din doi părinţi şi se înlocuiesc părinţii prin cei doi urmaşi pentru prelucrările din generaţia următoare. În algoritmul propus s-a utilizat însă o strategie diferită. După generarea urmaşilor, aceştia înlocuiesc un număr echivalent de indivizi din populaţia curentă, aleşi dintre cei cu viabilitatea cea mai redusă. Această strategie permite supravieţuirea soluţiilor mai bune de-a lungul unui mare număr de generaţii. Soluţiile generate prin operatorii genetici concurează pentru cel puţin o generaţie. Această metodă de selecţie a populaţiei următoare asigură performanţe corespunzătoare 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, încă în etapa de plasare sunt necesare măsuri speciale pentru a se asigura rutabilitatea circuitelor. Funcţia de cost utilizată optimizează diferite metrici, care includ lungimea totală a interconexiunilor, ca şi măsuri ale rutabilităţii plasării. Pentru calculul lungimii interconexiunilor s-a utilizat metoda semi-perimetrului. În general, minimizarea lungimii interconexiunilor are ca efect plasarea apropiată a celulelor în centrul circuitului, ceea ce va conduce aproape sigur la conexiuni care nu pot fi rutate. Pentru a se asigura rutabilitatea plasării, ideea este de a se utiliza un număr de celule libere pentru rutare. Aceste celule sunt alocate în cadrul procesului de plasare.

În timp ce minimizarea lungimii totale a interconexiunilor reduce numărul resurselor de rutare necesare în mod global, nu poate asigura faptul că semnalele vor putea fi rutate local, datorită resurselor limitate de rutare. Pentru rezolvarea acestei probleme, au fost adăugate două componente la funcţia de cost. Componenta de rutabilitate locală atribuie o penalizare acelor situaţii în care se poate determina faptul că o conexiune nu poate fi rutată utilizând resurse locale de rutare. Fiecare funcţie necesită două sau trei intrări, din care numai una poate fi furnizată printr-o magistrală locală sau expres [7]. De aceea, funcţiile cu două intrări trebuie să recepţioneze una din intrări de la celulele vecine, iar funcţiile cu trei intrări trebuie să recepţioneze două din intrări de la aceste celule. Există patru celule adiacente care pot furniza aceste intrări, iar funcţia de rutabilitate locală testează dacă semnalele de intrare necesare sunt fie prezente în aceste celule, fie există suficiente resurse de rutare disponibile astfel încât ele pot fi rutate. 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 datorită unei concentrări ridicate a funcţiilor într-o zonă a circuitului. Aceasta se realizează prin considerarea unor ferestre de celule (de 3×3 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. Ridicarea la pătrat este necesară pentru a se penaliza acele situaţii în care intrările utilizate au o concentraţie ridicată. Pragul este necesar pentru ca o logică de dimensiuni reduse să nu fie răspândită în cadrul circuitului.

Se examinează fiecare fereastră din circuitul FPGA, astfel încât ferestrele se suprapun. Este permisă deplasarea ferestrelor dincolo de limitele circuitului, pentru ce-

lulele virtuale aflate dincolo de marginile circuitului presupunându-se un număr de intrări utilizate egal cu media globală. În cazul în care nu s-ar permite deplasarea ferestrelor dincolo de limitele circuitului, sau s-ar presupune că celulele virtuale nu au intrări utilizate, un mare număr de celule ar fi concentrate la marginile circuitului. Similar, dacă s-ar presupune că celulele virtuale ar avea toate intrările utilizate, ar fi evitată alocarea celulelor de la margine.

Descrierea formală a algoritmului genetic pentru plasare este prezentată în Figura 4.29.

```
Algorithm Plasare_Genetică;
        /* Np  - dimensiunea populației      */
        /* Ng  - numărul de generații        */
        /* Nu  - numărul de urmaşi */
        /* Cr - rata de încrucişare          */
        /* Ir  - rata de inversiune          */
        /* Mr - rata de mutație              */
begin
        Se citeşte lista de conexiuni a circuitului;
        Se generează o populație inițială de Np configurații de plasare;
        for j = 1 to Np do
                Se evaluează Fitness (Populație [j]);
        endfor;
        Nu = Cr * Np;
        for (i = 1 to Ng) do
                for (j = 1 to Nu) do
                        /* Se selectează părinții pentru reproducție cu o probabilitate
                           proporțională cu valoarea viabilității */
                        (x, y) ← Sel_părinți;
                        /* Se aplică operatorul de încrucişare */
                        urmaş [j] ← Generare_urmaş (x, y);
                        Se aplică cu rata Mr operatorul de mutație asupra
                        configurațiilor selectate;
                        Se aplică cu rata Ir operatorul de inversiune asupra
                        configurațiilor selectate;
                endfor;
                for j = 1 to Np do
                        Se evaluează Fitness (Populație [j]);
                endfor;
                Se determină viabilitatea medie a populației;
                Se selectează Np configurații pentru următoarea generație, configurațiile
                cu viabilitate ridicată având o probabilitate mai mare de selecție;
        endfor;
        Se returnează configurația cu viabilitatea cea mai ridicată;
end.
```

Figura 4.29. Algoritm genetic pentru plasare.

Timpul de execuţie τ al algoritmului este o funcţie de dimensiunea populaţiei N_p, numărul de generaţii N_g, numărul de celule n, şi valorile asignate parametrilor genetici.

$$\tau(n, N_p, N_g) = t_{init}(n, N_p) + N_g \cdot (t_{rep}(n, N_p) + t_{eval}(n, N_p)) + t_{out}(n) \tag{4.97}$$

unde

t_{init} timpul necesar pentru citirea listei de conexiuni a circuitului şi pentru generarea populaţiei iniţiale;

t_{rep} timpul necesar pentru execuţia tuturor operaţiilor genetice (pentru o generaţie) şi pentru crearea unui set de noi indivizi;

t_{eval} timpul necesar pentru evaluarea funcţiei de cost pentru fiecare nou individ şi selecţia indivizilor care vor supravieţui în următoarea generaţie;

t_{out} timpul necesar pentru scrierea soluţiei finale pe disc.

Timpul cu ponderea cea mai mare este t_{eval}, iar $N_g \cdot t_{eval}$ este în jur de 75% din timpul total τ.

4.6.3 Rezultate experimentale

Algoritmul de plasare propus care utilizează secvenţa propusă a liniilor de tăietură 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*).

Experimentele au fost efectuate după cum urmează. Lista de conexiuni a circuitelor de test a fost convertită din formatul *EDIF* (*Electronic Design Interchange Format*) în formatul utilizat de programul de mapare tehnologică. Circuitele au fost apoi mapate tehnologic pentru circuitul FPGA *Atmel 6002*. Algoritmul de plasare a fost aplicat asupra listelor de conexiuni obţinute după maparea tehnologică.

În Tabelul 4.1 s-a comparat suma dimensiunilor tăieturilor pentru toate liniile de tăietură aplicate pentru algoritmul propus şi algoritmul tradiţional. Această sumă este o măsură a lungimii totale a conexiunilor atunci când estimarea se realizează prin metoda semiperimetrului. Comparaţia s-a efectuat pentru două cazuri. Primul caz este cel în care nu se urmăreşte echilibrarea numărului de conexiuni din cele două partiţii (*PONDERE* = 0), caz indicat în tabel prin *P* = 0. Al doilea caz este cel în care se realizează şi echilibrarea numărului de conexiuni, termenul corespunzător din funcţia de cost având aceeaşi pondere cu termenul pentru minimizarea lungimii conexiunilor (*PONDERE* = 1), caz indicat în tabel prin *P* = 1. Valorile s-au obţinut prin rularea algoritmilor de 20 de ori, calculându-se media sumei tăieturilor pentru aceste rulări (valorile fiind rotunjite la numere întregi).

Tabelul 4.1. Media sumei tăieturilor pentru algoritmul de plasare propus şi pentru algoritmul tradiţional.

	P = 0			P = 1		
Circuit	**Alg. propus**	**Alg. tradiţional**	**Reducere**	**Alg. propus**	**Alg. tradiţional**	**Reducere**
b1	13	19	31.5 %	15	17	11.7 %
c17	8	10	20.0 %	8	10	20.0 %
cm138a	23	38	39.4 %	28	38	26.3 %
con1	17	30	43.3 %	17	28	39.2 %
decod	44	69	36.2 %	51	59	13.5 %
majority	11	17	35.2 %	11	15	26.6 %
tcon	45	60	25.0 %	41	51	19.6 %
x2	38	51	25.4 %	42	46	8.6 %

Din Tabelul 4.1 rezultă că 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 (*P* = 0). În acest caz, reducerea medie este de 32 %. Pentru cazul în care se urmăreşte şi echilibrarea numărului de conexiuni (*P* = 1), reducerea obţinută este cuprinsă între 8.6 % şi 39.2 %, reducerea medie fiind de 20.6 %. Faptul că reducerea obţinută este mai mare în cazul în care nu se urmăreşte echilibrarea numărului de conexiuni este justificat, deoarece singura metrică în acest caz este dimensiunea tăieturii.

În Tabelul 4.2 s-a comparat valoarea maximã a tãieturii pentru toate liniile de tãieturã aplicate pentru algoritmul propus şi algoritmul tradiøional. Comparaøia s-a efectuat pentru aceleaşi douã cazuri: $P = 0$ şi $P = 1$.

Tabelul 4.2. Valorile maxime ale tãieturilor pentru algoritmul de plasare propus şi pentru algoritmul tradiøional.

	P = 0			P = 1		
Circuit	**Alg. propus**	**Alg. tradiţional**	**Reducere**	**Alg. propus**	**Alg. tradiţional**	**Reducere**
b1	7	9	22.2 %	8	11	27.2 %
c17	4	4	0.0 %	4	6	33.3 %
cm138a	9	12	25.0 %	14	16	12.5 %
con1	9	12	25.0 %	8	10	20.0 %
decod	10	12	16.6 %	15	17	11.7 %
majority	7	10	30.0 %	9	14	35.7 %
tcon	10	11	9.0 %	14	17	17.4 %
x2	12	14	14.2 %	12	16	25.0 %

Conform Tabelului 4.2, prin algoritmul propus s-a obøinut o reducere a tãieturii maxime cu un procent de pânã la 30 % pentru cazul în care nu se urmãreşte echilibrarea numãrului de conexiuni. În acest caz, reducerea medie este de 17.75 %. Pentru cazul în care se urmãreşte şi echilibrarea numãrului de conexiuni, reducerea obøinutã este de pânã la 35.7 %, reducerea medie fiind de 22.85 %. Deci, prin algoritmul propus se obøine atât o reducere a sumei tãieturilor, cât şi o reducere a valorii maxime a tãieturii.

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 populaøiei şi a ratei de mutaøie asupra calitãøii rezultatelor în scopul determinãrii unor valori optime pentru acešti parametri. Calitatea rezultatelor a fost apreciatã pe baza lungimii totale a interconexiunilor. Experimentele au arãtat cã nu se obøin diferenøe semnificative a lungimii totale a interconexiunilor cu variaøia ratei de mutaøie, dar existã diferenøe în funcøie de dimensiunea populaøiei. Tabelele 4.3 - 4.11 indicã lungimea totalã a interconexiunilor în starea iniøialã şi cea finalã în funcøie de dimensiunea populaøiei (rata de mutaøie fiind constantã, $M_r = 0.05$), şi timpul de execuøie al algoritmului pentru un numãr de generaøii $N_g = 1000$.

Tabelul 4.3. Efectul dimensiunii populaøiei asupra lungimii totale a interconexiunilor pentru circuitul *b1*.

Dimens.	**Lungimetotală a conexiunilor**		**Timp de execuţie**
populaţie	**Iniţială**	**Finală**	**(s)**
20	770	425	11
40	742	379	22
60	726	300	37
80	726	267	55
100	726	268	78
120	723	310	106

Tabelul 4.4. Efectul dimensiunii populaţiei asupra lungimii totale a interconexiunilor pentru circuitul *c17*.

Dimens.	Lungime totală a conexiunilor		Timp de execuţie
populaţie	Iniţială	Finală	(s)
20	363	184	7
40	363	163	14
60	363	129	22
80	363	105	32
100	355	149	44
120	355	129	56

Tabelul 4.5. Efectul dimensiunii populaţiei asupra lungimii totale a interconexiunilor pentru circuitul *cm138a*.

Dimens.	Lungime totală a conexiunilor		Timp de execuţie
populaţie	Iniţială	Finală	(s)
20	1201	677	17
40	1154	738	38
60	1154	543	59
80	1154	554	88
100	1154	519	120
120	1154	571	160

Tabelul 4.6. Efectul dimensiunii populaţiei asupra lungimii totale a interconexiunilor pentru circuitul *con1*.

Dimens.	Lungime totală a conexiunilor		Timp de execuţie
populaţie	Iniţială	Finală	(s)
20	899	458	12
40	899	498	26
60	899	388	43
80	899	330	61
100	887	387	84
120	887	318	109

Tabelul 4.7. Efectul dimensiunii populaţiei asupra lungimii totale a interconexiunilor pentru circuitul *daio*.

Dimens.	Lungime totală a conexiunilor		Timp de execuţie
populaţie	Iniţială	Finală	(s)
20	901	450	11
40	734	343	24
60	734	399	39
80	734	293	56
100	734	334	78
120	734	401	102

Tabelul 4.8. Efectul dimensiunii populației asupra lungimii totale a interconexiunilor pentru circuitul *decod*.

Dimens.	Lungime totală a conexiunilor		Timp de execuție
populație	Inițială	Finală	(s)
20	2485	1364	34
40	2485	1301	80
60	2485	1338	134
80	2485	1289	199
100	2485	1317	279
120	2485	1497	376

Tabelul 4.9. Efectul dimensiunii populației asupra lungimii totale a interconexiunilor pentru circuitul *majority*.

Dimens.	Lungime totală a conexiunilor		Timp de execuție
populație	Inițială	Finală	(s)
20	494	250	8
40	339	219	16
60	339	161	26
80	339	140	38
100	339	142	52
120	339	121	68

Tabelul 4.10. Efectul dimensiunii populației asupra lungimii totale a interconexiunilor pentru circuitul *tcon*.

Dimens.	Lungime totală a conexiunilor		Timp de execuție
populație	Inițială	Finală	(s)
20	2267	1388	33
40	2264	1215	77
60	2264	1242	126
80	2264	1296	185
100	2264	1249	255
120	2172	1323	338

Tabelul 4.11. Efectul dimensiunii populației asupra lungimii totale a interconexiunilor pentru circuitul *x2*.

Dimens.	Lungime totală a conexiunilor		Timp de execuție
populație	Inițială	Finală	(s)
20	2258	1264	28
40	2201	1240	63
60	2201	1162	103
80	2192	1211	154
100	2192	1090	213
120	2192	1246	285

Din aceste tabele rezultă că cele mai bune valori s-au obținut pentru dimensiuni ale populației de 80 sau 100. Pe baza acestor experimente, au fost utilizate următoarele valori ale parametrilor algoritmului genetic: dimensiunea populației N_p = 80, rata de mutație M_r = 0.05, rata de inversiune I_r = 0.15, numărul de generații N_g = 1000. Rezultatele obținute pentru acești parametri sunt prezentate în Tabelul 4.12.

Tabelul 4.12. Lungimea totală a interconexiunilor şi timpul de execuţie pentru setul de circuite de test.

Circuit	Lungime totală a conexiunilor		Timp de execuţie (s)
	Iniţială	Finală	
b1	726	267	55
c17	363	105	32
cm138a	1154	554	88
con1	889	330	61
daio	734	293	56
decod	2485	1289	199
majority	339	140	38
tcon	2264	1296	185
x2	2192	1211	154

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. Au fost prezentate diferite tehnici pentru estimarea lungimii totale a interconexiunilor pentru o plasare dată. Au fost descrise şi alte funcţii de cost utilizate, ca minimizarea tăieturii maxime şi minimizarea densităţii maxime.

Pentru rezolvarea problemei de plasare s-a utilizat un număr mare de tehnici euristice. Metodele bazate pe partiţionare implică aplicarea recursivă a unui algoritm de partiţionare, de obicei algoritmul Kernighan-Lin sau algoritmul de călire simulată. Deşi metoda de călire simulată poate obţine soluţia optimă globală, timpul de calcul necesar pentru circuite de dimensiuni mari este foarte mare. Pentru a elimina dezavantajul complexităţii ridicate a algoritmului de călire simulată, au fost propuse mai multe tehnici de creştere a vitezei pentru acest algoritm, dintre care au fost descrise cele propuse de Lam şi Delosme, respectiv Mallela şi Grover.

O altă metodă care a fost descrisă utilizează partiţionarea ierarhică pe baza tăieturii proporţionale. Circuitul este descompus mai întâi într-un arbore de grupuri prin aplicarea recursivă a metodei de partiţionare pe baza tăieturii proporţionale. În etapa a doua, fiecare grup este considerat ca o super-componentă, şi se aplică metoda de călire simulată pentru acestea. În etapa a treia, se integrează soluţiile subproblemelor prin utilizarea metodei ferestrelor mobile. Prin această metodă s-a obţinut o anumită reducere a dimensiunii maxime a tăieturii comparativ cu cea obţinută prin programul TimberWolfSC, şi o reducere a timpului de execuţie.

O posibilitate de rezolvare a problemei de plasare este transformarea acesteia într-o problemă de optimizare numerică. A fost descrisă o metodă numită plasare controlată de forţe, în care problema de plasare este redusă la soluţionarea unui sistem de ecuaţii liniare pentru a determina locaţiile de echilibru (coordonatele ideale x, y) ale celulelor. Dezavantajul acestor metodelor numerice este că ele necesită un timp de calcul foarte mare.

Metodele spectrale au fost utilizate şi pentru problema de plasare. În cazul unei asemenea metode, prin utilizarea unei funcţii liniare se obţine o plasare de calitate mai bună din punct de vedere a lungimii conexiunilor. Utilizarea unei funcţii obiectiv cuadratice are însă ca efect obţinerea unui număr mai redus de conexiuni foarte lungi faţă de cazul unei funcţii obiectiv liniare. A fost descrisă o metodă de plasare liniară propusă de Li care utilizează o funcţie obiectiv spectrală constând dintr-un compromis între funcţia cuadratică şi cea liniară, ceea ce permite folosirea avantajelor oferite de ambele funcţii.

Dintre metodele neconvenøionale de plasare, a fost prezentatã plasarea prin algoritmi paraleli şi plasarea prin reøele neuronale artificiale. A fost descrisã implementarea paralelã a unui algoritm de plasare care se bazeazã pe metoda de cãlire simulatã, care are aceleaşi proprietãøi de convergenøã ca şi algoritmul serial. Algoritmul utilizeazã douã moduri de paralelizare, în funcøie de valoarea temperaturii. A fost prezentatã o soluøie a problemei de plasare elaboratã de Yu, care a utilizat reøelele neuronale Hopfield. Acesta a modificat soluøia propusã de Hopfield şi Tank pentru rezolvarea problemei comis-voiajorului.

În acest capitol s-au propus doi algoritmi de plasare pentru circuitele FPGA cu resurse limitate de rutare, în particular pentru circuitele FPGA *Atmel*. Acești algoritmi au ca obiectiv asigurarea rutabilitãøii circuitelor. Primul algoritm se bazeazã pe algoritmul de partiøionare pe baza tãieturii minime propus în capitolul 3. Algoritmul de plasare propus utilizeazã bipartiøionarea a cãrei funcøie obiectiv urmãreşte 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. În cadrul plasării convenøionale bazate pe tãietura minimã, se alege linia de tãieturã poziøionatã în centrul regiunii curente. Aceastã alegere are ca rezultat o dimensiune a tãieturii relativ ridicatã pentru liniile de tãieturã din apropierea centrului, ceea ce are ca efect creşterea necesarului de resurse de rutare din aceastã zonã. În secvenøa propusã, în zona centralã liniile de tãieturã sunt aplicate în primele etape ale procesului de bipartiøionare. Rezultatele experimentale aratã cã prin aplicarea acestei secvenøe de tãieturã se obøine atât o reducere a dimensiunii maxime a tãieturii, cât şi 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 cunoştinøele noastre nefiind publicat pânã în prezent un algoritm genetic pentru plasarea acestor circuite. În cadrul algoritmului, strategia de selecøie a pãrinøilor încearcã sã evite convergenøa prematurã a algoritmului la un optim local. În algoritmul propus, strategia de înlocuire este diferitã de cea a algoritmilor tradiøionali. Dupã generarea urmaşilor, aceştia înlocuiesc un numãr echivalent de indivizi din populaøia curentã, aleşi dintre cei cu viabilitatea cea mai redusã. Aceastã strategie permite supravieøuirea soluøiilor mai bune de-a lungul unui mare numãr de generaøii.

Pentru facilitarea rutării, algoritmul alocã un numãr de celule libere în cadrul procesului de plasare, în scopul utilizării acestor celule pentru rutare. Algoritmul utilizeazã o funcøie de cost care optimizeazã diferite metrici, care cuprind atât lungimea interconexiunilor, cât şi mãsuri ale rutabilitãøii plasării. Funcøia de cost cuprinde o componentã de estimare a rutabilitãøii locale, cât şi o componentã de echilibrare a densitãøii. Algoritmul genetic o fost comparat cu un algoritm de plasare pe baza metodei de cãlire simulatã, rezultatele experimentale arãtând cã în cazul algoritmului genetic timpul de execuøie este mai redus, la o calitate a soluøiei care este apropiatã de cea obøinutã prin metoda de cãlire simulatã.

Contribuøiile acestui capitol sunt urmãtoarele:

- Elaborarea şi testarea unui algoritm de plasare pe baza bipartiøionării, a cãrei funcøie obiectiv urmãreşte minimizarea lungimii interconexiunilor simultan cu distribuøia echilibratã a conexiunilor din cele douã porøiuni.
- 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.
- Elaborarea şi testarea unui algoritm genetic pentru plasarea circuitelor FPGA, având ca obiectiv atât reducerea lungimii totale a interconexiunilor, cât şi asigurarea rutabilitãøii circuitului.