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

În acest capitol se prezintă un sistem CAD conceput și implementat pentru proiectarea sistemelor numerice utilizând circuitele FPGA din seria *Atmel* 6000. Specificaøia de intrare este o descriere în limbajul de descriere *ABEL*. Această descriere este compilată într-un set de ecuaøii cu ajutorul compilatorului sistemului de dezvoltare *Easy-ABEL*. Din setul de ecuaøii se generează o reprezentare internă a sistemului numeric. Apoi, sistemul CAD execută etapele de mapare tehnologică, plasare și rutare, și generează un fișier pentru configurarea circuitului FPGA. În final, structura circuitului configurat poate fi vizualizată în mod grafic.

6.1 Structura generală a sistemului CAD

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

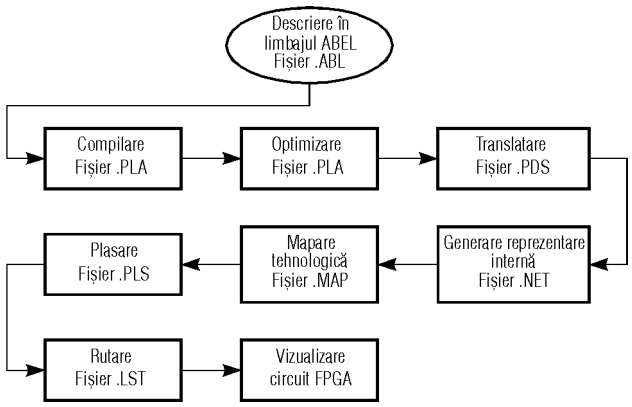

Figura 6.1. Structura generală a sistemului CAD.

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

Modulele implementate ale sistemului CAD pornesc de la fiøierul PDS. Pe baza acestuia se generează reprezentarea internã a sistemului sub forma unui graf. Pentru transformarea ecuaøiilor într-o formã corespunzãtoare circuitului FPGA, se executã etapa de mapare tehnologicã. Ecuaøiile sunt adaptate pentru a fi implementate cu ajutorul celulelor logice ale circuitului, cu optimizarea numãrului de celule necesare. Etapa urmãtoare este cea de plasare, în care se selectează locaøii specifice pentru fiecare celulã logicã. Modulul de rutare efectuează alocarea resurselor de rutare în scopul interconectãrii celulelor logice, generându-se un fiøier de configurare al circuitului. În final, structura circuitului configurat poate fi vizualizatã în mod grafic.

6.2 Descrierea proiectului

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

Descrierile pot fi introduse fãrã a øine cont de arhitectura circuitului care va fi utilizat pentru implementare. Descrierile independente de arhitecturã (care nu conøin declaraøii de dispozitive øi declaraøii de numere de pini) trebuie sã fie mai cuprinzãtoare decât cele specifice pentru o anumitã arhitecturã. Presupunerile care pot fi efectuate dacã se specificã un circuit particular nu sunt posibile dacã nu este specificat nici un circuit.

Un fiøier sursã *ABEL-HDL* poate fi împãrøit în mai multe pãrøi independente numite module. Fiecare modul conøine o descriere logicã completã. Deoarece fiecare modul este delimitat de un început øi un sfârøit, se pot combina mai multe fiøiere sursã, care conøin module, pentru a forma o descriere completã a unui sistem într-un fiøier sursã.

Un modul poate fi divizat în 5 secøiuni.

1) *Antetul*

Poate conøine urmãtoarele declaraøii:

- *Declaraøia Module*. Defineøte începutul unui modul. Acestei declaraøii trebuie sã îi corespundã o declaraøie *End*. Indicã dacã vor fi utilizate argumente formale (pentru apelul modulului cu parametri).
- *Declaraøia Options*. Se poate utiliza pentru a controla procesarea fiøierului sursã de cãtre compilator. Conøine opøiuni care pot fi specificate din linia de comandã sau din meniuri.
- *Declaraøia Title*. Indicã un antet care va apare în fiøierul document øi cel de programare al circuitului.

2) *Secøiunea de declaraøii*

Cuvântul-cheie *Declarations* permite declararea, în orice parte a fiøierului, a unor constante, atribute, macrouri, sau a unui tip de circuit. Declaraøiile sunt valabile în modulul respectiv. Fiecare modul trebuie sã aibã cel puøin o secøiune de declaraøii.

Existã mai multe tipuri de declaraøii.

- *Device*. Indicã un circuit, care se asociază cu un identificator de circuit.

- *Pin* şi *Node*. Declarã semnale utilizate, şi asociazã, opøional, numere de pini şi/sau de noduri la aceste semnale. Asocierea trebuie utilizatã de obicei numai atunci când se utilizeazã modulul *Fuseasm* sau *JEDSim*.
- *ISTYPE*. Cu aceastã declaraøie se pot asocia atribute semnalelor în cadrul declaraøiilor *Pin* şi *Node*. Aceastã declaraøie defineşte atributele semnalelor pentru circuitele cu caracteristici programabile sau în cazul în care nu s-a specificat un tip de circuit sau numãr de pin/nod pentru un semnal.

3) *Descrierea logicii*

Pentru descrierea sistemului se pot utiliza unul sau mai multe din urmãtoarele elemente: ecuaøii, tabele de adevãr, diagrame de stare.

3.a. *Ecuaøii*

Declaraøia EQUATIONS defineşte începutul unui grup de ecuaøii care specificã funcøiile logice ale circuitului. Sintaxa declaraøiei este urmãtoarea:

```
EQUATIONS
       [WHEN condiţie THEN] [!]
       element = expresie;
       [ELSE ecuaţie];
```

sau

```
       [WHEN condiţie THEN] ecuaţie;
       [ELSE ecuaţie];
```

3.b. *Tabele de adevãr*

Specificã ieşirile ca funcøii de diferite combinaøii ale intrãrilor, sub formã tabelarã. Descrierea conøine un antet care indicã formatul tabelei, urmat de tabela însãşi. Sintaxa generalã este:

```
TRUTH_TABLE (intrări → ieşiri)
             intrări → ieşiri;
```

3.c. *Diagrame de stare*

Pentru specificarea diagramelor de stare este necesarã utilizarea declaraøiei STATE_DIAGRAM, care defineşte maşina de stare, şi a declaraøiilor IF-THEN-ELSE, CASE şi GOTO, care definesc funcøionarea maşinii de stare.

4) *Secøiunea vectorilor de test*

Vectorii de test sunt opøionali şi se utilizeazã pentru verificarea funcøionalitãøii proiectului logic. Vectorii de test specificã operarea funcøionalã aşteptatã a unui circuit prin definirea ieşirilor sale ca o funcøie de intrãri. Definirea se realizeazã cu declaraøia TEST_VECTORS, cu sintaxa generalã:

```
TEST_VECTORS (intrări → ieşiri)
             (valori_de_intrare → valori_de_ieşire);
```

5) *Sfârşitul modulului*

Declaraøia END defineşte sfârşitul modulului:

```
END nume_modul;
```

În Figura 6.2 se prezintã un exemplu de descriere de intrare pentru sistemul CAD, descriere realizatã în limbajul *ABEL*.

```
module actlow

TITLE   'Functional example of active low output signals

"inputs
     clock  pin 1;
     in1    pin 2;
     in2    pin 3;

"outputs
     !out1   pin 23 ISTYPE 'reg,buffer'; "active low signal
      out2   pin 22 ISTYPE 'reg,buffer'; "active high signal
     !out3   pin 21 ISTYPE 'reg,invert'; "active low signal
      out4   pin 20 ISTYPE 'reg,invert'; "active high signal

" Constant assignments
     x, c, H, L  =  .x., .c., 1, 0 ;

" Set Declaration
     outputs  =  [ out1, out2, out3, out4 ];

equations
     out1   :=   in1  &  in2 ;
     out2   :=   in1  &  in2 ;
     out3   :=   in1  &  in2 ;
     out4   :=   in1  &  in2 ;
     outputs.clk  =  clock ;

test_vectors
     ([ clock, in1, in2 ] -> [ out1, out2, out3, out4 ])
      [   0  ,  x ,  x  ] -> [  x  ,  x  ,  x  ,  x   ];
      [   c  ,  0 ,  0  ] -> [  L  ,  L  ,  L  ,  L   ];
      [   c  ,  1 ,  1  ] -> [  H  ,  H  ,  H  ,  H   ];
      [   c  ,  0 ,  1  ] -> [  L  ,  L  ,  L  ,  L   ];

end
```

Figura 6.2. Exemplu de descriere în limbajul *ABEL*.

6.3 Compilarea şi optimizarea descrierilor

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

Operaøiile principale efectuate de compilator sunt urmãtoarele:

- testează și indicã erorile de sintaxã din fișierul sursã;
- efectuează conversia diagramelor de stare și a tabelelor de adevãr în ecuaøii;
- translatează vectorii de test și crează un fișier cu vectorii de test (*.tmv);
- expandează macrourile;
- efectuează conversia ecuaøiilor care conøin seturi în ecuaøiile echivalente fãrã seturi;
- înlocuiește operatorii cu operaøii echivalente utilizând numai operatorii NOT, AND, OR și XOR;

- combinã ecuaøiile care determinã asignãri multiple la acelaši identificator;
- efectueazã reducerea logicã a ecuaøiilor pe baza regulilor algebrei booleene;
- transferã ecuaøiile într-un fišier cu formatul standard PLA (*.ttn).

Prin apelarea modulului *PLASim* se aplicã intrãri ecuaøiilor din fišierul PLA, utilizând fišierul vectorilor de test generat de compilator. Aceasta se efectueazã deci înaintea alegerii unui circuit ši a asignãrii pinilor. Se genereazã un fišier cu rezultatul simulãrii (*.smn).

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

Selecøiile cu *polaritatea aleasã automat* permit generarea ecuaøiilor atât pentru polaritatea pozitivã, cât ši pentru cea negativã. Dacã sunt disponibile ambele seturi de ecuaøii, modulele de generare a fišierului de programare pentru circuit ši de adaptare pentru un anumit dispozitiv vor alege setul cel mai potrivit pentru circuitul selectat.

Selecøiile cu *polaritatea fixã* limiteazã modulul de optimizare la polaritatea specificatã cu 'pos' sau 'neg' în fišier.

Sunt descrise în continuare opøiunile pentru optimizare [64].

- *Modul implicit.* Se va utiliza setarea din fišier sau modul implicit al programului pentru minimizare. Se executã minimizarea pin-la-pin *Espresso*, cu polaritatea automatã, cu excepøia cazului în care s-a declarat un nume de circuit care începe cu 'F' (arhitecturã FPLA). Pentru acest circuit, modulul *PLAOpt* executã minimizarea *Espresso* pe grup, cu polaritate fixã.

- *Minimizare pe pini.* Opøiunea *bypin* produce rezultate cele mai bune pentru circuite PAL ši satisfãcãtoare pentru circuite FPLA. Se executã minimizarea pentru fiecare iešire, independent de alte ieširi. Se poate indica polaritatea automatã sau cea fixã.

- *Minimizare pe grup.* Opøiunea group va minimiza ecuaøiile pentru un circuit FPLA unde un singur termen produs poate fi utilizat de cãtre mai multe ieširi.

- *Sinteza automatã a bistabilelor D/T.* Opøiunea -reduce dt determinã utilizarea optimã a emulãrii bistabilelor D ši T. Modulul *PLAOpt* va utiliza o combinaøie de bistabile D ši T pentru a obøine implementarea cea mai redusã. Aceastã opøiune poate salva muløi termeni produs la circuitele care dispun de porøi SAU EXCLUSIV, acestea, împreunã cu bistabilele de tip D fiind utilizate pentru emularea bistabilelor de tip T.

- *Fãrã optimizare.* Se reunesc toate ecuaøiile compilate într-un singur fišier de tip PLA, fãrã minimizarea acestora.

6.4 Generarea reprezentării interne

Specificaøia de intrare pentru implementarea unui sistem numeric utilizând sistemul CAD propus ši implementat este o descriere într-un limbaj de descriere hardware de nivel înalt. Limbajul utilizat este *ABEL-HDL* (*ABEL Hardware Description Language*). Din fišierul sursã conøinând descrierea de intrare se genereazã o descriere intermediarã utilizând compilatorul sistemului de dezvoltare *Easy-ABEL*. Aceastã descriere este în formatul unui fišier .PDS, conøinând ecuaøiile sistemului descris.

Fišierul .PDS conøine douã secøiuni principale:

- declaraøii (titlul proiectului, numele circuitului, lista de pini, lista nodurilor etc)
- ecuaøii, care descriu funcøionarea circuitului.

Fiecare ecuaøie din fiøierul .PDS este analizatã øi se generează un graf, ale cãrui noduri sunt componente logice de bazã (porøi ØI/SAU cu douã intrãri, inversoare, bistabile D, buffere cu trei stãri).

Depinzând de funcøionalitatea circuitului, semnalele memorate într-un bistabil pot avea una din urmãtoarele extensii:

.CLKF Aceastã extensie indicã un semnal care este conectat la intrarea CLK a unui bistabil D. Extensia corespunde extensiei *.clk* din limbajul *ABEL*.

.TRST Aceastã extensie indicã un semnal care controlează ieøirea (în acest caz, un pin de ieøire cu trei stãri). Extensia corespunde extensiei *.oe* din limbajul *ABEL*.

.SETF, .RSTF Aceste extensii indicã semnale care efectuează setarea sau resetarea unui bistabil D. Extensiile corespund extensiei *.pr* din limbajul *ABEL*.

În Figura 6.3 se prezintã ecuaøiile generate în urma compilãrii descrierii pentru circuitul *Actlow* øi a translatãrii în formatul PDS.

```
TITLE ACTLOW

;PINLIST (Highest pin number = 24)
    CLOCK IN1 IN2 NC NC NC NC NC NC NC NC NC NC NC NC NC NC NC NC
OUT4
    OUT3 OUT2 OUT1 NC

EQUATIONS

/OUT1 := (IN1 * IN2);
OUT1.CLKF = (CLOCK);
OUT2 := (IN1 * IN2);
OUT2.CLKF = (CLOCK);
/OUT3 := (IN1 * IN2);
OUT3.CLKF = (CLOCK);
OUT4 := (IN1 * IN2);
OUT4.CLKF = (CLOCK);
```

Figura 6.3. Ecuaøiile obøinute prin compilarea descrierii circuitului *Actlow* øi translatarea în formatul PDS.

După construirea grafului fiecãrei ecuaøii, noul graf trebuie adãugat grafului general al circuitului. O etapã importantã este eliminarea redundanøelor. În locul unor grafuri separate multiple, aceste grafuri sunt combinate într-unul singur, dacã este posibil, încercându-se minimizarea numãrului de noduri, øi în consecinøã, complexitatea circuitului. Această operaøie este numitã eliminarea nodurilor duplicate øi este executatã la analiza fiecãrei ecuaøii.

În final, pe baza listei de pini, se identificã pinii de intrare øi de ieøire. Programul efectuează distincøia dintre pinii de ieøire øi nodurile interne. Un nod intern este un semnal utilizat numai în interiorul circuitului. Programul generează o matrice, în care a_{ij} = 1 dacã există o conexiune între nodul *i* øi *j* (o muchie de la *i* la *j* sau de la *j* la *i* în graful circuitului).

În Figura 6.4 se prezintã fiøierul care conøine reprezentarea internã a circuitului *Actlow* sub forma unui graf. Numãrul din primul câmp al fiecãrei linii indicã un nod al grafului, tipul nodului apare după caracterele T:, iar lista de noduri succesoare ale acestui nod este indicat după caracterele S:.

```
1#T:f0#S:15#
3#T:f3#S:1#
4#T:I#L:IN1#S:5#
5#T:f1#S:14, 12, 9, 3#
6#T:I#L:IN2#S:5#
7#T:I#L:CLOCK#S:14, 12, 9, 3#
9#T:f3#S:16#
10#T:f0#S:17#
12#T:f3#S:10#
14#T:f3#S:18#
15#T:O#L:/OUT1#
16#T:O#L:OUT2#
17#T:O#L:/OUT3#
18#T:O#L:OUT4#
```

Figura 6.4. Fişierul cu reprezentarea circuitului *Actlow* sub forma unui graf.

6.5 Maparea tehnologicã

Algoritmii de mapare tehnologicã se bazează pe tehnici booleene de potrivire, deci de recunoaştere a echivalenøei între o porøiune a reøelei şi celulele din setul disponibil. În momentul execuøiei mapării tehnologice, se presupune cã optimizarea logicã a fost deja efectuatã.

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

Maparea tehnologicã implicã douã operaøii distincte: recunoaşterea echivalenøei logice între douã funcøii logice, şi determinarea setului optim de porøi echivalente din punct de vedere logic, ale cãror interconexiuni reprezintã circuitul original. Prima operaøie, numitã *potrivire*, implicã testarea echivalenøei şi asignarea intrãrilor. Testarea echivalenøei logice este NP-completã. Asignarea intrãrilor este de asemenea o operaøie complexã din punct de vedere computaøional. A doua operaøie, numitã *acoperire*, implicã gãsirea unei reprezentări alternative a unei reøele booleene utilizând elemente logice care au fost selectate dintr-un set restrâns [118].

Pentru sistemul CAD propus, maparea tehnologicã implementatã se bazează pe o metodã algoritmicã. Cele mai multe metode algoritmice divid problema de mapare tehnologicã în sub-probleme. Mai întâi, reøeaua booleanã este partiøionatã într-o interconexiune de sub-reøele cu câte o singurã ieşire, cu proprietatea cã fiecare vârf intern are un fan-out unitar. Apoi, fiecare sub-reøea este descompusã într-o interconexiune de funcøii cu douã intrări (ŞI, SAU, ŞI-NU, SAU-NU). Fiecare sub-reøea este modelatã printr-un graf aciclic direcøionat. În final, fiecare graf este acoperit printr-o interconexiune de celule de bibliotecã, pentru a se obøine circuitul final.

În continuare se descriu principalele operaøii efectuate în cadrul mapării tehnologice.

Partiøionarea este o operaøie care transformã problema de mapare tehnologicã pentru reøele cu ieşiri multiple într-o secvenøã de subprobleme implicând reøele cu câte o singurã ieşire. Partiøionarea este executatã în timpul fazei iniøiale, ca şi pentru îmbunãtãøirea iterativã a unei reøele mapate. Partiøionarea iniøialã este realizatã prin gruparea vârfurilor în sub-reøele cu câte o singurã ieşire, cu proprietatea cã fiecare muchie de ieşire a unui vârf intern reconverge la sau înaintea vârfului de ieşire al sub-reøelei.

Partiøionarea este utilizatã de asemenea pentru izolarea porøiunii combinaøionale a unei reøele de elementele secvenøiale şi de celulele de I/E. Dupã etapa de par-

tiøionare, circuitul este reprezentat printr-un set de circuite combinaøionale care pot fi modelate prin *grafuri subiect* (reøele booleene cu o singurã ieşire).

Decompoziøia este aplicatã asupra fiecãrui graf subiect dupã partiøionare. În urma acestei operaøii se obøine un graf subiect echivalent, în care fiecare vârf este o funcøie de bazã, de exemplu o funcøie ŞI/SAU/ŞI-NU/SAU-NU cu douã intrãri. Decompoziøia reprezintã o soluøie de mapare pentru biblioteci care includ funcøiile de bazã, ceea ce este cazul majoritãøii bibliotecilor. Aceastã operaøie creşte de asemenea granularitatea reøelei, ceea ce este avantajos pentru etapa de acoperire.

În acest moment, circuitul logic care trebuie mapat a fost partiøionat în grafuri subiect [Γ_1, ... Γ_k], care au fost descompuse. Se notează cu Γ_f un graf subiect al cãrui vârf cu o singurã ieşire este v_f. Se considerã acoperirea unui graf subiect Γ_f care optimizează un anumit criteriu de cost. În acest scop se utilizează noøiunile de *grup* şi *funcøie de grup*.

Un *grup* este un subgraf conectat al grafului subiect Γ_f, având un singur vârf cu o singurã ieşire. Se caracterizează prin adâncimea sa (calea direcøionatã cea mai lungã la v_i) şi numãrul de intrãri. Funcøia de grup asociatã este funcøia booleanã obøinutã prin comasarea expresiilor booleene asociate cu vârfurile subgrafului într-o singurã funcøie booleanã. Toate grupurile posibile având ca rãdãcinã vârful v_j din Γ_f şi funcøiile lor sunt notate cu $\{\kappa_{i.1}, ..., \kappa_{i.n}\}$.

Algoritmul de *acoperire* încearcã sã potrivească fiecare funcøie de grup cu un element din bibliotecã. O acoperire este un set de grupuri asociate cu elemente de bibliotecã, care acoperã graful subiect. Costul unei acoperiri este calculat prin adunarea costului grupurilor corespunzãtoare variabilelor din funcøia de grup $\kappa_{i.k}$ la costul elementului de bibliotecã, pentru orice permutare a variabilelor sale.

Algoritmul de mapare tehnologicã utilizat porneşte de la reprezentarea internã a circuitului. Primele etape ale mapãrii tehnologice sunt partiøionarea şi decompoziøia. Aceste etape au fost implementate în cadrul etapei de generare a reprezentãrii interne, reøeaua logicã generatã având proprietãøile cerute. Reøeaua logicã este deja descompusã, conøinând numai componentele logice de bazã [16].

De aceea, singura etapã care trebuie implementatã este acoperirea, care include şi etapa de potrivire booleanã. Pentru acoperirea reøelei, trebuie sã se ia în considerare setul de configuraøii logice disponibile în cadrul circuitului FPGA *Atmel* 6002.

Un aspect important de care trebuie sã se øinã cont este cã maparea tehnologicã va fi urmatã de etapele de plasare şi rutare. Dacã numãrul de celule logice generate în cadrul mapãrii tehnologice este prea mare, aceasta va creşte în mod considerabil complexitatea etapei de plasare.

În cadrul etapei de rutare, trebuie sã se øinã cont de capacitatea maximã a canalelor de rutare ale circuitului FPGA. În programul de mapare tehnologicã, pentru a se asigura rutabilitatea circuitului, se efectuează duplicarea celulelor a cãror fan-out este mai mare decât aceastã capacitate maximã.

Din cele de mai sus, rezultã cã trebuie realizat un echilibru între:

- generarea unui numãr cât mai mic de celule în scopul reducerii complexitãøii etapei de plasare, şi
- generarea unui numãr suficient de celule pentru a se asigura rutabilitatea circuitului mapat.

Algoritmul de mapare tehnologicã este prezentat în Figura 6.5 [16].

```
Algorithm Mapare_tehn_Atmel;
begin
        forall nodurile de intrare
                Se mapează nodul la o celulă de intrare;
        endfor;

        forall bufferele de ieşire cu trei stări
                Se determină succesor (nod);
                Se mapează bufferul de ieşire cu trei stări împreună cu
                        succesorul acestuia la  o celulă de ieşire;
        endfor;

        forall nodurile de ieşire rămase
                Se mapează nodul la o celulă de ieşire;
        endfor;

        forall bistabilele D
                Se mapează nodul la o celulă;
                /* ceasul este distribuit pe coloane */
                Se asignează celulele la aceeaşi coloană;
        endfor;

        forall celulele de ieşire mapate
                recurse (sub-arbore (celulă));
                potrivire_booleană (nod, element_bibliotecă);
                nod = nod_curent;
                while (fan-out (nod) > CAPACIT_MAX) do
                        duplicate (nod, nod_nou);
                        update_link (nod); /* se actualizează toate legăturile */
                                           /* la succesorii şi legăturile de la */
                                           /* predecesorii nodului curent */
                        update_link (nod_nou);
                endwhile;
        endfor;
end.
```

Figura 6.5. Algoritmul de mapare tehnologicã pentru circuitul FPGA *Atmel* 6002.

Rezultatul mapării tehnologice pentru circuitul *Actlow* este prezentat în Figura 6.6.

```
#1#T:8#S = Bo: 2 Ai Li Bi ;
#2#T:33#;
#3#T:21#S = Ao, Lo: 1 Li ;
#4#T:32#S = Ao, Lo, Bo: 5 Ai ;
#5#T:28#S = Bo: 3 Ai 8 Ai 7 Ai 6 Ai ;
#6#T:21#S = Ao, Lo: 14 Ai Li Bi ;
#7#T:21#S = Ao, Lo: 12 Li ;
#8#T:21#S = Ao, Lo: 11 Ai Li Bi ;
#9#T:32#S = Ao, Lo, Bo: 5 Bi ;
#10#T:32#S = Ao, Lo, Bo: 3 Ci 8 Ci 7 Ci 6 Ci ;
#11#T:33#;
#12#T:8#S = Bo: 13 Ai Li Bi ;
#13#T:33#;
#14#T:33#;
```

Figura 6.6. Fişierul cu rezultatul mapării tehnologice pentru circuitul *Actlow*.

6.6 Plasarea celulelor

Pentru plasare s-a implementat un algoritm care utilizeazã partiøionarea pe baza tãieturii minime. Fiind dat un circuit reprezentat printr-un graf, fiecare vârf reprezentând o celulã logicã şi fiecare muchie reprezentând o conexiune, algoritmul partiøionează în mod recursiv circuitul şi suprafaøa de plasare, până când se ajunge la un graf cu un singur nod, care va fi plasat într-o regiune a suprafeøei. Unul din obiectivele fiecãrei etape de bipartiøionare este minimizarea dimensiunii tãieturii, cu restricøia ca fiecare porøiune sã conøinã acelaşi numãr de noduri, sau un numãr apropiat.

În cazul algoritmilor de plasare tradiøionali, 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. Pe baza acestor considerente, în cadrul funcøiei de cost s-a luat în considerare nu numai dimensiunea tãieturii, ci şi distribuøia interconexiunilor din cele douã porøiuni ale bipartiøiei.

Algoritmul de plasare implementat utilizează bipartiøionarea a cãrei funcøie obiectiv urmãreşte minimizarea lungimii interconexiunilor simultan cu minimizarea diferenøei între numãrul de conexiuni din cele douã porøiuni. Această diferenøã, numită număr de dezechilibru, este măsura care urmăreşte distribuøia echilibratã a conexiunilor din cele douã porøiuni. În cadrul algoritmului de plasare se aplicã în mod recursiv bipartiøionarea până când se ajunge la porøiuni care conøin o singurã celulã a circuitului. Liniile de tãieturã se aplicã în mod alternativ, pe orizontalã şi pe verticalã.

S-a utilizat urmãtoarea funcøie de cost:

$$Cost = Dim\_tăietură + PONDERE \times Dezechilibru \tag{6.7}$$

unde *PONDERE* este o constantã. Dacã această constantã este setatã la zero, algoritmul este acelaşi cu cel convenøional. Prin setarea corespunzãtoare a ponderii se poate controla importanøa echilibrării numărului de conexiuni.

Pe lângã o procedurã eficientã de partiøionare, este necesarã o strategie 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. Această 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. Deoarece linia de tãieturã consideratã este în apropierea liniei de tãieturã din centru, regiunile intersectate au o suprafaøã redusã. De aceea, numărul nodurilor din aceste regiuni este redus. În consecinøã, numărul perechilor posibile care pot fi interschimbate în procesul de bipartiøionare este limitat. De obicei, aceasta are ca rezultat o dimensiune a tãieturii relativ ridicatã pentru liniile de tãieturã din apropierea centrului, din cauza numărului mic de mutări posibile.

Din cele de sus rezultã cã, pentru a se reduce dimensiune tãieturii în apropierea centrului, în această zonă liniile de tãieturã trebuie aplicate în primele etape ale procesului de bipartiøionare. La implementarea algoritmului de plasare s-a øinut cont de aceste considerente. Liniile de tãieturã orizontale şi verticale sunt aplicate alternativ în cadrul algoritmului.

Algoritmul de plasare care minimizează dimensiunea tãieturii şi simultan echilibrează numărul de conexiuni din cadrul partiøiilor este prezentat în Figura 6.7 [13] [14].

```
Algorithm Plasare_Atmel (N, 2n, G, PONDERE);
        /* N este suprafaţa de plasare                                          */
        /* 2n este numărul de celule care trebuie plasate                       */
        /* G este graful circuitului                                            */
        /* PONDERE indică importanţa echilibrării numărului de conexiuni        */
begin
    (N1, N2) = DIVIZ_SUPRAF (N);
    G1 = PART_INIT (G, n);
    G2 = G - G1;
    bestGAIN = ∞;
    while bestGAIN > 0 do
        nr_dezechilibru = DEZECHILIBRU (n);
        bestGAIN = 0; bestk = 0;
        for i = 1 to 2n do Blocat (i) = 0; endfor;
        for k = 1 to n do
            gain (k) = 0; GAIN (k) = 0;
            for all v_i ∈ G1 AND Blocat (i) = 0 do
                for all v_j ∈ G2 AND Blocat (j) = 0 do
                    gain_cut = D_i + D_j - 2 c_ij;
                    nr_dezechilibru_nou = nr_dezechilibru - E_i - I_i + E_j + I_j;
                    gain_dez = | nr_dezechilibru - nr_dezechilibru_nou |;
                    if (gain_cut + PONDERE * gain_dez) > gain (k) then
                        gain (k) = gain_cut + PONDERE * gain_dez;
                        max1 (k) = i; max2 (k) = j;
                    endif;
                endfor;
            endfor;
            Blocat (max1 (k)) = Blocat (max2 (k)) = 1;
            for i = 1 to 2n do
                if v_i ∈ G1 AND Blocat (i) = 0 then
                    D_i = D_i - c_i,max2(k) + c_i,max1(k)
                endif;
                if v_i ∈ G2 AND Blocat (i) = 0 then
                    D_i = D_i - c_i,max1(k) + c_i,max2(k);
                endif;
            endfor;
            GAIN (k) = GAIN (k-1 ) + gain (k);
            if GAIN (k) > bestGAIN then
                bestk = k; bestGAIN = GAIN (k);
            endif;
        endfor;
        for k = 1 to bestk do
            (G1, G2) = INTERSCH (V1, V2, v_max1(k), v_max2(k));
        endfor;
    endwhile;
    Plasare_Atmel (N1, n, G1, PONDERE);
    Plasare_Atmel (N2, n, G2, PONDERE);
end.
```

Figura 6.7. Algoritmul de plasare pentru circuitele FPGA *Atmel*.

6.7 Rutarea circuitului

Algoritmul de rutare implementat executã simultan rutarea globalã şi cea detaliatã. Se eliminã astfel dezavantajul legat de împãrţirea problemei de rutare în douã

subprobleme, cea de rutare globalã şi de rutare detaliatã, care conduce la rezultate globale care nu sunt optime. Un alt avantaj al acestei metode este cã estimarea preliminarã a rutãrii globale poate fi imediat corectatã. De asemenea, algoritmul poate lua în considerare efectele secundare pe care le au deciziile de rutare luate pentru o conexiune asupra celorlalte conexiuni.

Pentru rutarea semnalelor pe distanøe scurte, algoritmul utilizeazã celule libere în locul magistralelor. Magistralele sunt rezervate pentru rutarea semnalelor pe distanøe mai mari (mai mult de cinci celule), pentru semnale cu trei stãri, sau pentru semnale cu un fan-out ridicat.

Algoritmul utilizeazã magistrale expres ori de câte ori este posibil. Aceste magistrale nu sunt conectate direct la celule, de aceea ele au o încãrcare capacitivã mai redusã şi sunt mai rapide decât magistralele locale. De asemenea, prin utilizarea unei magistrale expres în locul uneia locale, magistrala localã este eliberatã pentru alte conexiuni. Totuşi, înlocuirea unei magistrale locale printr-o magistralã expres nu este posibilã în anumite cazuri: la conectarea directã la o celulã; la utilizarea unui semnal bidirecøional; la efectuarea unei întoarceri cu 90°.

Pentru creşterea performanøelor, algoritmul limiteazã numãrul segmentelor magistralelor locale prin care trece un semnal şi utilizeazã repetoarele numai dacã este necesar. Ramificarea semnalului de pe o magistralã expres la magistrala localã la fiecare repetor poate fi beneficã atunci când valoarea fanout-ului este mai mare decât opt, sau dacã semnalul trece prin mai mult de un repetor.

Fiecãrui segment de interconectare al circuitului i se asociazã douã costuri: costul distanøei (C_d), care reflectã întârzierile de rutare asociate cu segmentul de interconectare, şi costul competiøiei (C_c), care contorizeazã legãturile care se aflã în competiøie pentru acelaşi segment. Fiecãrei cãi din lista de conexiuni i se asociazã de asemenea douã costuri: suma costului distanøelor (S_d) pentru segmentele cãii, şi suma costului competiøiei (S_c) pentru segmentele cãii.

Pentru reducerea necesarului de memorie şi al timpului de execuøie, algoritmul nu ia în considerare toate posibilitãøile de interconectare din cadrul circuitului într-o singurã etapã. De aceea se utilizeazã o metodã iterativã. În prima etapã se considerã numai acele cãi posibile pentru o conexiune care corespund costului C_d minim. Dacã aceste cãi sunt în conflict cu conexiunile deja rutate, algoritmul continuã cãutarea pornind de la costul cãii eşuate.

Algoritmul de rutare este descris în Capitolul 5 (Figura 5.48).

În prima etapã a algoritmului, se construieşte graful de conectivitate pe baza structurii circuitului FPGA, iar apoi se construieşte graful de rutare pe baza rezultatelor obøinute dupã maparea tehnologicã şi plasare. Prin legãturi directe se desemneazã acele legãturi care nu implicã utilizarea unei magistrale.

În urmãtoarea etapã, algoritmul încearcã divizarea conexiunilor în legãturi directe în cazul în care acestea nu trebuie sã treacã prin mai mult de cinci celule. Pentru fiecare conexiune, algoritmul determinã distanøa minimã a cãii de rutare, şi memoreazã toate alternativele de rutare într-o listã a conexiunilor.

Algoritmul poate efectua douã tipuri de optimizãri: din punct de vedere al spaøiului ocupat şi din punct de vedere al vitezei. Pentru optimizarea din punctul de vedere al *spaøiului*, algoritmul sortează mai întâi conexiunile după numărul alternativelor posibile de rutare (numărul căilor din graf, utilizând costul S_c), astfel încât conexiunile care au un număr mai redus de alternative vor fi mai prioritare. După selectarea unei conexiuni prin această procedură de sortare, algoritmul utilizează o funcøie de cost pentru evaluarea costului fiecărei căi disponibile, şi alege calea cu costul S_d minim.

```
[0 4]
F : f0 , 0
0,4 AiV 0,3 AoE 0
0,4 AoS 1,4 AiN 0
1

[1 3]
F : f5:OUT1 , 0
1,3 AiV 1,2 AoE 0
1,3 BoN 0,3 BiS 0
1

[1 2]
F : f3 , 0
1,2 AiV 1,1 AoE 0
1,2 AoE 1,3 AiV 0
1,2 AiN 0,2 AoS 0
1

[0 0]
F : I:IN1 , 0
0,0 AoS 1,0 AiN 0
1

[1 1]
F : f1 , CT2 , 0
1,1 AiV 1,0 AoE 0
1,1 BoV 1,0 BiE 0
1,1 AoS 2,1 AiN 0
1,1 LS1 Lx1 Ly1 Lx3 3,2 LS3 0
1,1 AoE 1,2 AiV 0
1,1 AiN 0,1 AoS 0
1

[0 1]
F : I:IN2 , 0
0,1 AoS 1,1 AiN 0
1

[0 2]
F : I:CLOCK , CT2 , 0
0,2 AoS 1,2 AiN 0
0,2 LS0 Lx0 Ly2 4,2 LE2 0
0,2 LS0 Lx0 Ly2 2,2 LE2 0
0,2 LE2 Ly2 3,2 LE2 0
1

[2 3]
F : f5:OUT2 , 0
2,3 AiV 2,2 AoE 0
2,3 AoE 2,4 AiV 0
1

[2 2]
F : f3 , 0
2,2 AiV 2,1 AoE 0
2,2 AoE 2,3 AiV 0
2,2 LE2 0
1

[3 3]
F : f0 , CT1 , 0
3,3 LV3 0
3,3 AoE 3,4 AiV 0
1

[3 1]
F : f5:OUT3 , 0
3,1 LN3 Lx3 Ly3 3,3 LV3 0
3,1 BiV 3,2 BoE 0
1

[3 2]
F : f3 , 0
3,2 BoV 3,1 BiE 0
3,2 LS3 0
3,2 LE2 0
1

[4 3]
F : f5:OUT4 , 0
4,3 AiV 4,2 AoE 0
4,3 AoE 4,4 AiV 0
1

[4 2]
F : f3 , 0
4,2 AiV 4,1 AoE 0
4,2 AoE 4,3 AiV 0
4,2 LE2 0
1

[1 4]
F : O:/OUT1 , 0
1,4 AiN 0,4 AoS 0
1

[2 4]
F : O:OUT2 , 0
2,4 AiV 2,3 AoE 0
1

[3 4]
F : O:/OUT3 , 0
3,4 AiV 3,3 AoE 0
1

[4 4]
F : O:OUT4 , 0
4,4 AiV 4,3 AoE 0
1

[0 3]
F : L , 0
0,3 BiS 1,3 BoN 0
0,3 AoE 0,4 AiV 0
1

[1 0]
F : L , 0
1,0 AoS 2,0 AiN 0
1,0 AoE 1,1 AiV 0
1,0 BiV 1,1 BoE 0
1,0 AiN 0,0 AoS 0
1

[2 0]
F : L , 0
2,0 AoS 3,0 AiN 0
2,0 AiN 1,0 AoS 0
1

[2 1]
F : L , 0
2,1 AoE 2,2 AiV 0
2,1 AiN 1,1 AoS 0
1

[3 0]
F : L , 0
3,0 AoS 4,0 AiN 0
3,0 AiN 2,0 AoS 0
1

[4 0]
F : L , 0
4,0 AoE 4,1 AiV 0
4,0 AiN 3,0 AoS 0
1

[4 1]
F : L , 0
4,1 AiV 4,0 AoE 0
4,1 AoE 4,2 AiV 0
1
```

Figura 6.8. Lista de conexiuni obținutã în urma rutării circuitului *Actlow*.

Pentru optimizarea din punctul de vedere al *vitezei*, algoritmul sortează mai întâi conexiunile după lungimea lor (utilizând costul S_d). Astfel, va determina ca liniile lungi să aleagă magistralele expres, care sunt mai rapide. Dintre toate conexiunile posibile, se alege cea cu costul S_c minim.

În etapa de rutare, algoritmul selectează alternativele de rutare cu costul S_c sau S_d minim. În etapa de rerutare, se încearcă găsirea altor căi posibile, pe baza grafului de conectivitate actualizat. În această etapă, în graful de conectivitate se marchează punctele de conectivitate utilizate de conexiunile deja rutate. Celula care generează semnalul conexiunii este de asemenea marcat în graful de conectivitate. Astfel, în etapa de rerutare alternativele de rutare care pornesc de la un punct de conectivitate care este utilizat pentru alte conexiuni vor fi luate în considerare.

Funcția de cost utilizatã în cazul optimizării pentru rutabilitate are rolul de a selecta o cale de rutare care va avea un efect negativ redus asupra conexiunilor rã-

mase, din punct de vedere al rutabilitãøii. Aceastã funcøie împiedicã selectarea cãilor care conøin segmente de interconectare pentru care existã un numãr mare de cereri. Probabilitatea de utilizare a unei muchii depinde de numãrul alternativelor posibile la utilizarea acestei muchii. Deci, costul unei muchii *e* pentru care existã alte *j* alternative (muchii în paralel cu *e*) va fi invers proporøional cu numãrul alternativelor.

În Figura 6.8 se prezintã lista de conexiuni obøinutã în urma rutãrii circuitului *Actlow*.

6.8 Concluzii

În acest capitol s-a prezentat un sistem CAD conceput şi implementat pentru proiectarea sistemelor numerice utilizând circuitele FPGA din seria *Atmel* 6000. Sistemul CAD a fost implementat în limbajul C++, funcøionând sub sistemul de operare Windows 95. Sistemul a fost conceput pe baza metodologiei de proiectare care a fost propusã în cadrul tezei, şi integreazã rezultatele obøinute în studiul etapelor de partiøionare, plasare şi rutare. În plus, sistemul realizeazã maparea tehnologicã pentru circuitul FPGA utilizat. A fost realizatã de asemenea o interfaøã cu sistemul de proiectare *EasyABEL*, interfaøã care permite descrierea sistemului numeric în limbajul *ABEL*, compilarea descrierii într-un set de ecuaøii şi optimizarea acestora.

Specificaøia de intrare este o descriere în limbajul de descriere *ABEL*. Aceastã descriere este compilatã într-un set de ecuaøii, sub forma unui fişier în formatul PLA. Fişierul generat este optimizat, minimizându-se logica în scopul reducerii termenilor produs. Se pot introduce diferite opøiuni de optimizare şi de minimizare logicã. În urma optimizãrii se obøine de asemenea un fişier în formatul PLA, care este translatat într-un fişier în formatul PDS. Acest fişier conøine ecuaøiile sistemului care trebuie implementat.

Pe baza fişierului PDS se generează reprezentarea internã a sistemului sub forma unui graf. Pentru transformarea ecuaøiilor într-o formã corespunzãtoare circuitului FPGA, se executã apoi etapa de mapare tehnologicã. Ecuaøiile sunt adaptate pentru a fi implementate cu ajutorul celulelor logice ale circuitului, cu optimizarea numãrului de celule necesare. În continuare se executã etapa de plasare, în care se selectează locaøii specifice pentru fiecare celulã logicã. Urmează etapa de rutare, în care se efectuează alocarea resurselor de rutare în scopul interconectãrii celulelor logice, generându-se un fişier de configurare al circuitului. În final, structura circuitului configurat poate fi vizualizatã în mod grafic.

Contribuøia acestui capitol o reprezintã conceperea şi implementarea unui sistem CAD pentru proiectarea sistemelor numerice utilizând circuitele FPGA din seria *Atmel* 6000. Sistemul se bazează pe metodologia de proiectare care a fost propusã în cadrul tezei, şi integreazã rezultatele obøinute în studiul etapelor de partiøionare, plasare şi rutare. În plus, sistemul conøine un program de mapare tehnologicã pentru circuitele FPGA *Atmel* şi o interfaøã care permite utilizarea limbajului de descriere *ABEL*.