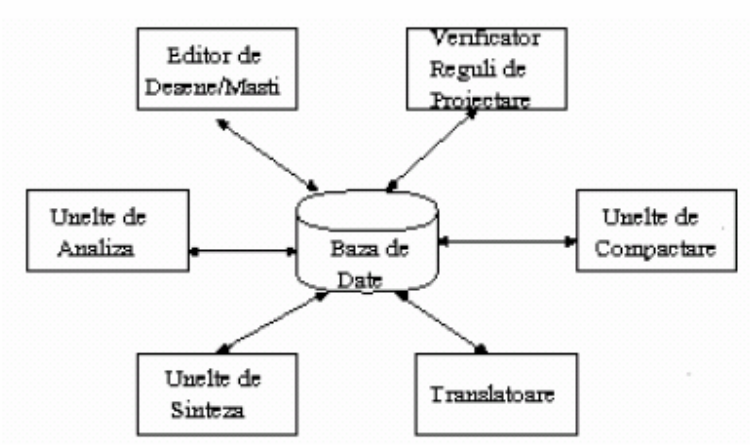
Capitolul 7. Unelte de proiectare.

Perfectionarea tehnicilor de fabricatie, cat si reducerea dimensiunilor dispozitivelor au condus la realizarea unor sisteme numerice mari si complexe pe o singura pastila. Complexitatea crescanda a circuitelor integrate comerciale curente se datoreaza partial si existentei unelteIor de Poiectare Asistata de Calculator (PAC). In acest mod proiectantul nu mai are de-a face cu lucrarile/task-urile de nivel coborat, putandu-se concentra pe aspectele de nivel inalt ale proiectarii circuitelor integrate. In acest capitol se vor prezenta pe scurt cateva din uneltele disponibile pentru PAC, cat si algoritmii utilizati.. Pentru cea mai mare parte a uneltelor PAC discutate in acest capitol se vor examina aspectele de implementare de nivel coborat.
Un mediu idealizat de proiectare este aratat mai jos. Fiecare unealta folosita in proiectarea circuitelor integrate partajeaza o baza comuna de date. Interfatarea cu alte unelte de proiectare, care utilizeaza o baza de date diferita se realizeaza prin translatarea informatiei de proiectare in formate de interschimb, care reprezinta limbaje standard suportate de catre cei mai multi ofertanti de unelte PAC.



Motivul pentu care schema de sus este idealizata este acela ca, de regula, o parte dintre ofertantii de unelte sunt angajati pentru anumite aspecte ale proiectarii. In acest caz, adesea, iesirea oferita de catre o unealta catre alta o reprezinta o lista a componentelor/conexiunilor.

7.1. Descrierea proiectului.

Cele mai importante unelte folosite in mediul de proiectare de circuite integrate sunt uneltele de descriere si bazele lor de date asociate. Aceste unelte sunt utilizate pentru crearea, editarea si intretinerea informatiei de proiectare, indiferent ca este vorba de masti, de conectivitate sau informatie despre comportament.

In aceasta sectiune se vor examina cateva programe pentru descrierea proiectului, cat si tehnicile referitoare la bazele de date asociate.
Indiferent de mediul de descriere a proiectului, se impune realizarea unei interfete prietenoase cu utilizatorul pentru toate programele. Acestea vor fi programele pe care proiectantul le va folosi cel mai frecvent.

7.1.1. Editoare de masti.

Editoarele de masti sunt programe folosite pentru a descrie, modifica si intretine geometriile desenelor.

Cele mai multe programe de desenare sunt capabile sa deseneze forme simple, care sunt in mod curent utilizate in proiectarea circuitelor integrate. De asemenea, cele mai multe editoare de masti contin elemente specifice incorporate, pentru proiectarea acestora. Este foarte importanta interactiunea cu verificatoarele de reguli de proiectare, cat si cu alte unelte de proiectare.

Caracteristicile dorite pentru un editor de masti sunt urmatoarele:

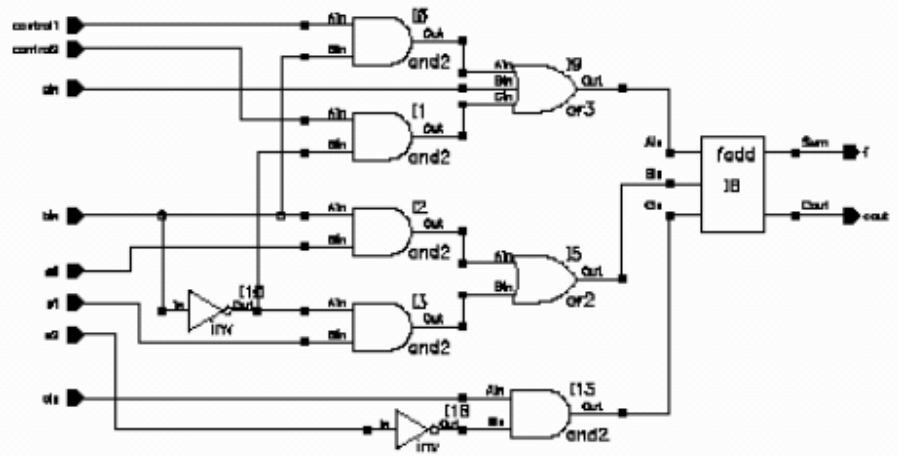
- Programul trebuie sa suporte geometriile des intalnite in desenele de masti. Tipic, cele mai multe proiecte de circuite integrate au de-a face cu poligoane simple avand unghiuri multiple de 45 grade.
- Programul trebuie sa suporte proiectarea ierarhica. Proiectarea ierarhica permite managementul unor proiecte mari, cat si incurajarea reutilizarii unor module proiectate anterior.
- In vederea proiectarii blocurilor, trebuie sa existe suport pentru diferite niveluri de abstractizare.

Exista numeroase moduri pentru exprimarea informatiei privind proiectarea. Pentru geometria mastilor se folosesc programe de vizualizare a mastilor/desenelor, iar pentru informatia de conectivitate se utilizeaza listele de componente/conexiuni (net list) si schemele. Gestiunea diferitelor module de vizualizare este esentiala.

- Trebuie intretinuta informatia referitoare la conectivitate, cat si la formele geometrice. Este mult mai convenabil a avea de-a face cu dispozitive de tipul tranzistoarelor si a structurilor, cum ar fi contactele, ca unitati recunoscute de catre program, decat cu reprezentari ale dispozitivelor sub forma de poligoane.
- Comenzile pentru editarea desenelor/mastilor trebuie sa fie capabile sa deplaseze obiecte, mentinand in acelasi timp conectivitatea. De exemplu, atunci cand este deplasat un tranzistor, toate conexiunile sale trebuie sa se deplaseze odata cu el.
- Programul trebuie sa ofere flexibilitate in ceea ce priveste interfata cu alte unelte de proiectare. Un editor de desene/masti, ca atare, nu prezinta o importanta prea mare. Verificatoarele de reguli de proiectare, extractoarele de circuit, simulatoarele si alte asemenea unelte, folosite in procesul de proiectare, trebuie sa fie cat mai strans legate de programul pentru desenarea mastilor.
- Programele trebuie sa fie independente de tehnologie. Progresele rapide in fabricarea circuitelor integrate vor face ca un program legat de un proces dat sa fie depasit inaite de a fi terminat. De asemenea, cei mai multi proiectanti au la dispozitie mai multe procese, iar invatarea unui nou program pentru fiecare ar fi nerealista.
- Vor fi permise celule parametrizate. Trebuie sa fie posibila crearea unor celule ale caror proprietati sa poata fi ajustate prin modificarea parametrilor externi. De exemplu, crearea unui inversor cu latimi ale tranzistoarelor ce pot fi specificate extern.
- Trebuie sa fie permise ambele metodologii de proiectare: “top-down” si “bottom-up”. Uneltele nu trebuie sa-l forteze pe proiectant sa foloseasca o anume metodologie; ele trebuie sa fie destul de flexible pentru a permite utilizarea unui numar mare de tehnologii.

In multe cazuri un proiectant de ASIC nu va folosi editorul de masti. El va fi preocupat de mai mult de descrierea schemei, de un limbaj de de descriere hardware sau de lista componentelor/conexiunilor Uneltele pentru desenare, pe de alta parte, sunt esentiale daca se urmareste realizarea unor produse de varf, cum ar fi memoriile si procesoarele. De asemenea, daca proiectul are in vedere viteza de lucru ridicata sau daca este vorba de circuite analogice, proiectantul va fi preocupat de detaliile desenelor mastilor. Planurile mastilor afecteaza atat suprafata ocupata, cat si performantele. In cazul circuitelor de foarte mare viteza, diafonia si cuplajele reprezinta probleme legate de desenele mastilor.

7.1.2. Descrierea schemei.
In cazul existentei unor unelte sofisticate pentru generarea mastilor, proiectantul nu va mai fi preocupat de geometria acestora.
Descrierea la nivel de schema poate fi utilizata pentru a introduce informatia de conectivitate din cadrul proiectului. Proiectantul realizeaza descrierea circuitului prin plasarea simbolurilor in proiect si prin conectarea corespunzatoare cu fire a porturilor simbolurilor date. In mod curent simbolurile utilizate in cadrul schemelor sunt macrocelulele de mari dimensiuni, porti din biblioteca de celule standard sau tranzistoare disponibile in tehnologia data. In figura care urmeaza se prezinta proiectul schemei unui tronson de Unitate Aritmetica Logica (UAL).



Un proiect ierarhizat poate fi descris prin crearea simbolurilor de reprezentare a blocurilor de nivel mai coborat si conectarea lor ca si in cazul portilor.
Dupa introducerea schemei, informatia de conectivitate intre componentele proiectului este extrasa prin urmarirea firelor la porturile simbolurilor. In aceasta etapa pot fi detectate erorile simple de proiectare, cum ar fi retelele neconectate. Descrierea la nivel de schema permite proiectantului sa extraga listele de componente/conexiuni fara sa creeze mastile. Aceasta ofera posibilitatea simularii proiectului inca in faza de proiectare. Proiectul poate fi extins pentru a include intarzieri generice, iar dupa plasare si rutare, parametrii privind intarzierea pot fi din nou introdusi in schema, pentru o simulare mai detaliata.

7.1.3. Limbaje de Descriere a Hardware-lui (LDH - HDL).
In locul descrierii informatiei privitoare la proiect folosind sisteme grafice interactive asociate cu editoare de masti si de scheme, adesea este mult mai convenabil sa se introduca informatia sub forma de text. LHD sunt utilizate pentru a descrie informatia referitoare la proiect in cazul sistemelor mari, pe baza unor constructii mult mai abstracte decat cele permise in cadrul descrierilor la nivel de scheme. LDH vor fi descrise mai tarziu in cadrul acestui capitol.

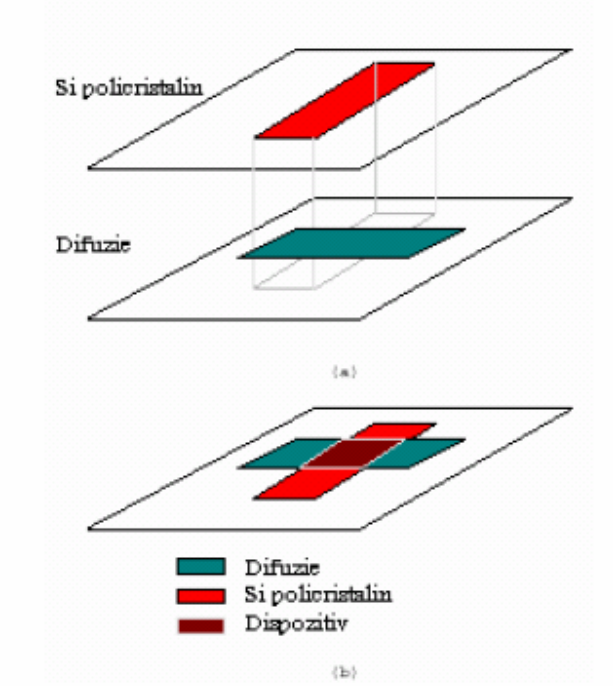
7.1.4. Bazele de date.
Oricaredintre programele folosite pentru introducerea proiectelor utilizeaza o baza de date pentru stocarea acestora. In mod ideal aceasta baza de date trebuie sa fie accesibila tuturor uneltelor de proiectare. Este mult mai convenabil sa se dispuna de o singura baza de date partajata de catre toate programele, decat sa existe o baza de date pentru fiecare program. In cazul de fata se recomanda ca proprietati:
- Baza de date trebuie sa utilizeze cat mai mult posibil sistemul de fisiere al sistemului de operare. In loc de a stoca toate proiectele intr-un singur fisier mare, care reprezinta biblioteca, fiecare proiect trebuie sa fie salvat intr-un fisier separat, biblioteca fiind, astfel, directorul care contine aceste

fisiere. Aceasta permite un management mai facil si mai consistent al bibliotecii de proiectare. Utilitarele sistemului de operare pot fi folosite pentru a gestiona baza de date.
- Baza de date poate stoca informatie in format binar sau text. Formatul binar este mai compact, iar cel de tip text este mai flexibil.

7.2. Unelte de desenare.
Programele pentru editarea desenelor mastilor sunt de cea mai mare importanta in proiectarea circuitelor integrate. Chiar daca se folosesc pentru descriere si introducere scheme sau LDH-uri, un editor de masti ramane inca necesar. In continuare se vor prezenta unele tehnici folosite pentru gestiunea geometriilor desenelor. De asemenea, se vor prezenta unele dintre programele existente pentru proiectarea circuitelor integrate. Exista, in acest sens, o multime de unelte disponibile, produse de catre Mentor Graphics si de catre Cadence. Capabilitatile unor asemenea unelte vor fi prezentate pe scurt.

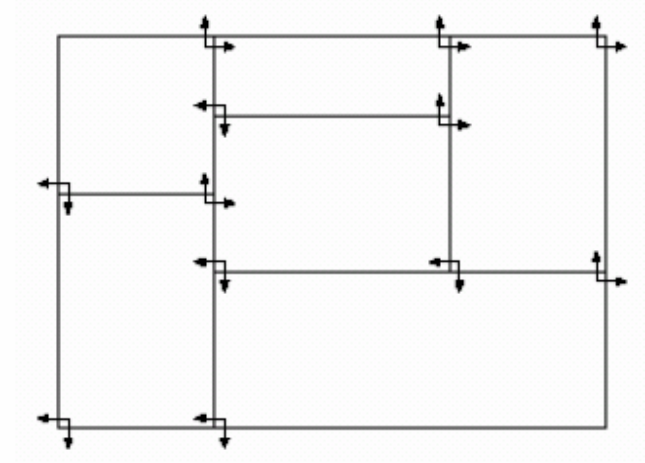
7.2.1. Reprezentarile planurilor mastilor.

7.2.1.1. Acoperirea.
Una dintre cele mai simple metode pentru reprezentarea geometriei planurilor se bazeaza pe acoperire.
Mastile unui circuit integrat sunt reprezentate ca un set de planuri, fiecare fiind acoperit cu forme care reprezinta geometria mastilor. In loc de a stoca masca prezenta in fiecare punct al proiectului, va fi pastrata o lista de forme care pot acoperi portiuni mari ale structurii.
Mai jos sunt prezentate doua modalitati prin care informatia despre masti poate fi acoperita. In figura a) fiecare nivel al mastilor este reprezentat intr-un plan diferit, iar in figura b) fiecare plan contine masti multiple. Acoperirile pot reprezenta niveluri individuale sau multiple.



O baza de date va stoca toate aceste acoperiri permitand executia unor programe de verificare a regulilor de proiectare si de generare a informatiilor privind mastile.

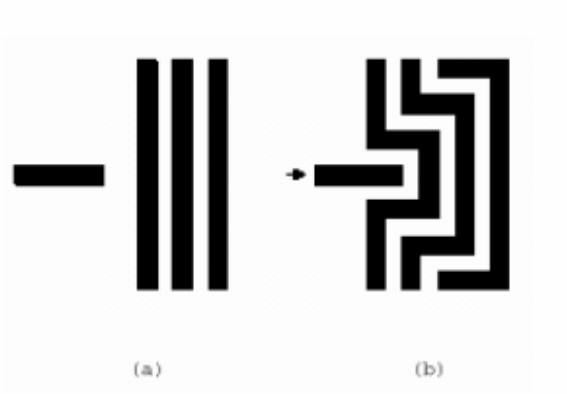
7.2.1.2 Alaturarea colturilor.
Lipirea colturilor reprezinta un algoritm de acoperire, care permite existenta intr-un plan numai a formelor care nu se suprapun. Spatiile libere sunt acoperite cu forme care nu reprezinta masti. Fiecare forma indica alte patru forme, dupa cum se vede din figura.



Cu toate ca aceasta metoda necesita mai mult spatiu de memorie decat alte tehnici, exista o serie de algoritmi eficienti pentru gasirea formelor vecine. Trebuie sa fie posibila, de exemplu, operatia de tip "rasturnare" (plow), ca in sistemul Magic.
Din punctul de vedere al proiectantului, reprezentarea interna a geometriei desenului nu este importanta. Este important sa se realizeze ca, acesta informatie trebuie stocata intr-o maniera care sa permita algoritmilor o eficienta manipulare a datelor.

Electric este un program pentru proiectarea circuitelor integrate, disponibil pentru universitati, si comercializat sub numele "*Electric*", de catre Electric Editors. In cadrul acestui pachet software, circuitele sunt reprezentate sub forma unei colectii de arce si noduri. Nodurile pot fi noduri simple predefinite, pentru elemente comune de circuit cum ar fi tranzistoarele, contactele, capacitorii, sau noduri mai complexe, care reprezinta celule create de catre utilizator. Fiecare nod poseda porturi care definesc directiile de acces extern, cat si straturile. Arcele sunt folosite pentru conectarea impreuna a nodurilor.
Una din caracteristicile deosebite ale pachetului *Electric* a constat in existenta facilitatii de verificare automata a regulilor de proiectare, pe masura elaborarii proiectului, ceea ce permite corectarea rapida a eventualelor erori.

7.2.2. Magic.
Magic este un program de desenare a mastilor de circuite integrate dezvoltat la Universitatea Berkeley, California. El are in cea mai mare parte caracteristici similare cu *Electric*, dar prezinta si unele particularitati. In continuare se prezinta cateva din caracteristicile pachetului *Magic*.
- Magic poseda un Verificator de Reguli de Proiectare (VRP) incremental.
- *Magic* dispune de un "rutor" de canal. Prin introducerea informatiei de conectivitate in fisier, *Magic* poate conecta proiectul dupa amplasarea celulelor.
- Magic permite operatia de "rasturnare", exemplificata mai jos, care asigura permite compactarea si expandarea proiectelor.



7.2.4. Cadence.

Cadence reprezinta un pachet industrial puternic, pentru proiectarea circuitelor integrate. *Cadence* ofera o gama larga de unelte de proiectare, bazate pe o platforma comuna. Aceste unelte acopera spectrul necesitatilor de proiectare cum ar fi: *amplasarea* si *rutarea, simularea, modelarea, sinteza* si *verificarea regulilor de proiectare*.

Spre deosebire de *Electric* si *Magic*, *Cadence* nu furnizeaza codul sursa pentru sistem. In schimb, platforma *Cadence* poseda un limbaj de programare denumit *SKILL*, care este similar cu *C* si *LISP*. Cu ajutorul acestui limbaj, capabilitatile sistemului pot fi extinse fara a mai fi necesara modificarea codului sursa pentru unelte. In mod similar modulul, care genereaza lista componentelor/conexiunilor este suficient de flexibil pentru a permite utilizatorului sa-si adauge interfetele proprii.

• *Cadence* nu este comercializat ca un singur program, ci ca o colectie de unelte, care pot fi adaugate la un sistem de proiectare de baza, la un nucleu.

• *Cadence* dispune de capabilitatile de *Amplasare* si *Rutare* pentru sinteza automata a planurilor mastilor, plecand de la scheme.

• Spre deosebire de *Electric* si *Magic*, *Cadence* nu permite operarea simultana a diferitelor module/unelte. De exemplu, nu este disponibil un VRP incremental.

• Ca optiuni, *Cadence* include: *descrierea simbolica a mastilor, compactarea, VRP, extragerea circuitului din masti, unelte pentru circuite analogice, generatoare de module.*

7.3. Unelte pentru generarea desenelor mastilor.

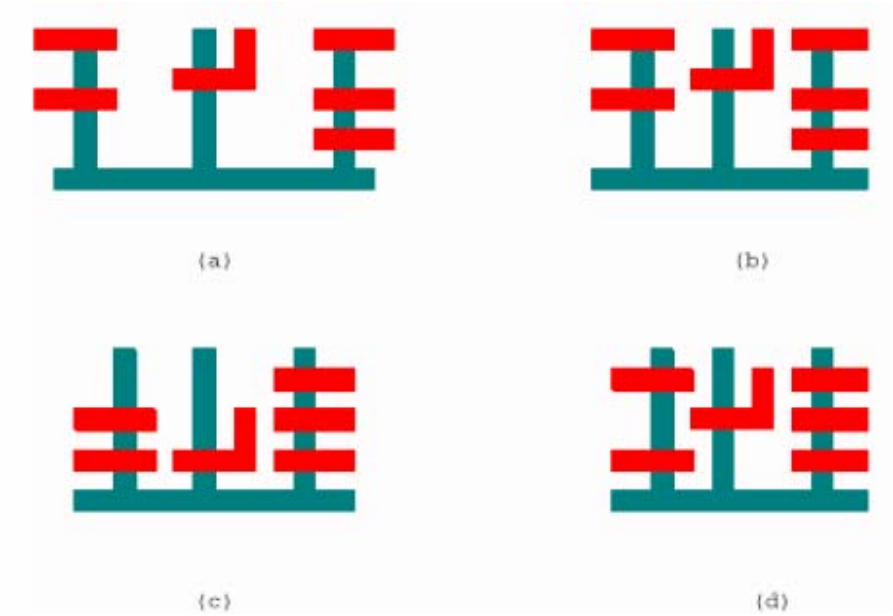
Pentru proiecte mici, proiectantul va dori sa elaboreze manual desenele mastilor. Pentru proiectele mari, cea mai mare parte a activitatilor poate fi automatizata, folosind diverse unelte. Pe langa reducerea timpului de proiectare, Uneltele de Proiectarea Automata (UPA) sunt mai putin afectate de de proiectare. Uneltele de generare nu sunt tot atat de bune ca si expertii umani. In orice caz timpul de proiectare poate fi redus pe seama dimensiunilor si a vitezei de lucru. In aceasta sectiune vor fi pe scurt, cativa algoritmi pentru proiectarea automata a mastilor.

7.3.1. Unelte de compactare.

Compactarea desenelor mastilor este importanta nu numai pentru cresterea densitatii circuitelor, ci si pentru reducerea capacitatilor parazite, asociate cu firele lungi. Desi compactarea se poate realiza pe intregul circuit, complexitatea problemei limiteaza compactarea numai la celule de dimensiuni reduse.

Uneltele de compactare sunt intens utilizate in conjunctie cu programele de generare a celulelor, care efectueza sinteza desenelor mastilor acestora. Aceste programe au tendinta de a crea desene de masti functionale, dar nu in mod necesar compacte.

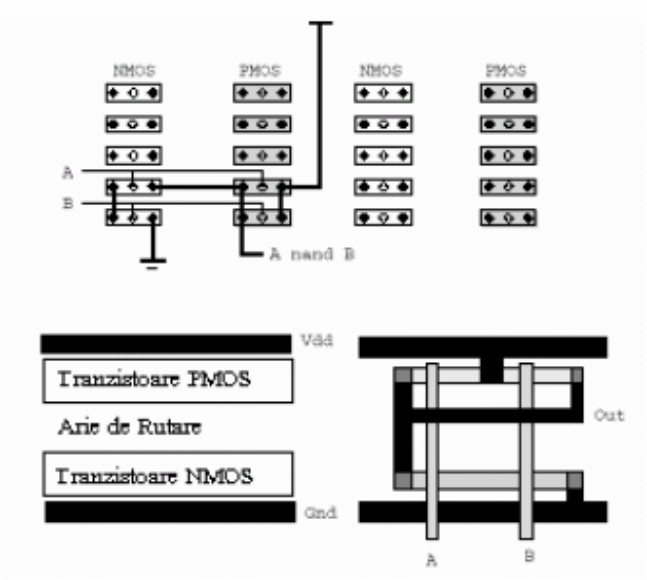
Compactarea unidimensiunala este cea mai simpla forma de algoritm de compactare. Mai intai desenul este compactat in directia ***x*** prin deplasarea orizontala a obiectelor pana la obtinerea separatiei minime.
Apoi desenul este compactat in directia ***y***. Compactarea in directiile ***x*** si ***y*** este repetata pana cand nu mai este posibila optimizarea. In figura de mai jos este dat un exemplu de compactare unidimensionala.



Desenul original este prezentat in figura (a). Dupa prima compactare pe directia ***x***, se obtine rezultatul din figura (b). Rezultatul compactarii dupa directia ***y*** este dat in figura (c). In figura (c) se prezinta rezultatul compactarii optime, care nu se poate obtine ca rezultat al compactarii dupa o singura dimensiune.
Compactarea bidimensionala incearca sa efectueze compactarea simultan in directiile ***x*** si ***y***. Este evident ca aceasta reprezinta o sarcina mult mai complexa necesitand, in general, euristici si algoritmi de Inteligenta Artificiala. (IA). De asemenea, au fost utilizati algoritmii nedeterministi, cum ar fi *calirea simulata* sau *algoritmii genetici.*
*Calirea simulata**) efectueaza optimizarea prin deplasarea aleatoare a formelor geometrice. Daca o deplasare imbunatateste proiectul ea va fi pastrata. Daca o deplasare degradeaza proiectul, modificarea este pastrata cu o probabilitate dependenta de degradare si de temperatura. Aceasta permite algoritmului sa se elibereze de minimele locale in cadrul optimizarii. Temperatura este scazuta pe masura desfasurarii algoritmului.
*Algoritmii genetici***) pornesc de la un ansamblu de solutii posibile, ansamblul genelor. Cele mai bune gene sunt copiate pentru ansamblul reuniune. Sunt selectate aleator perechi de gene, care isi schimba parti ale solutiilor lor. In acest moment se poate introduce o mica mutatie. Aceste gene reprezinta acum generatia curenta a ansamblului de gene. Copierea probabilista, reunirea si mutatia sunt continuate pana la atingerea unei solutii rezonabile. Ca si in cazul *calirii simulate* algoritmul este simplu desi scrierea programului, cat si obtinerea unei solutii acceptabile pot fi dificile.
In continuare se prezinta o serie de aspecte referitoare la compactare.
- Compactarea este dependenta de planul initial. Algoritmii de compactare nu vor pleca de la un plan oarecare pentru a produce planuri bune pentru masti. Este necesar un plan initial rezonabil.
- Insertia unor deviatii in firele de conectare poate imbunatati compactare. Aceasta creste numarul formelor geometruce in planul mastilor.

- Interschimbul de componente in locul deplasarii lor poate conduce la imbunatatirea compactarii.
- Trebuie sa se aibe grija sa nu se inlature spatiile introduse pentru motive bine determinate.
De exemplu, pentru a imbunatati fiabilitatea, un proiect doreste sa pastreze anumite dispozitive separate printr-o distanta mai mare decat distanta minima.
- In proiectarea ierarhica, deplasarea unui obiect poate afecta obiectele de la diferitele niveluri ale ierarhiei.
- Compactarea *non-mahhattan* introduce mai multe posibilitati de deplasare, care trebuie avute in vedere.
- Compactarea nu trebuie sa modifice separarile, care sunt esentiale pentru proiect. De exemplu, separarea traseelor/sinelor, in celulele standard, si a locatiilor porturilor, in celulele, care urmeaza sa fie suprapuse, trebuie sa fie conectate la locatii specifice.

7.3.2. Generarea celulelor.
Adesea este important sa se dispuna de mijloace automate pentru sinteza planurilor celulelor.
In continuare se vor prezenta diferite tehnici pentru generarea celulelor.
Marea de porti utilizeaza o arie de tranzistoare la care se adauga conexiuni pentru a realiza functiile logice dorite. Aceasta este o idee simpla dar nu foarte eficienta in ceea ce priveste aria ocupata.
Pentru crearea planurilor mastilor portilor poate fi utilizata o tehnica de sinteza de tipul *amplaseaza* si *ruteaza*. Aceasta este asemanatoare cu tehnica amplaseaza si ruteaza ce se va prezenta in sectiunea urmatoare, cu diferenta ca, in loc de porti, aici sunt amplasate tranzistoare. Tipic portile vor avea o structura similara. De exemplu, cele mai multe generatoare de celule CMOS vor folosi o structura de tipul celei de mai jos.



Tranzistoarele PMOS sunt plasate pe o singura linie sub traseul/sina de alimentare, iar tranzistoarele NMOS se gasesc pe o singura linie deasupra traseului de masa. Aria cuprinsa intre tranzistoarele PMOS si NMOS este utilizata pentru rutarea conexiunilor la porti ale tranzistoarelor complementare.
Pentru structurile regulate de tipul registrelor, tampoanelor/buffer-elor si numaratoarelor se folosesc generatoare de porti speciale. Aceasta tehnica este cel mai des utilizata pentru a realiza componente de inalta performanta cu structura repetitiva. Pentru fiecare poarta este necesar un generator unic. Generatorul poate dispune de posibilitati de scalare a tranzistoarelor, cat si de posibilitati privind optimizarea parametrilor.

In scopul reducerii ariei ocupate de planul mastilor, dupa sinteza acestora, se poate efectua operatia de compactare.

7.3.3. Minimizarea logica.

Tehnicile de minimizare logica au aparut inainte de proiectarea circuitelor integrate. Pentru gasirea implicantilor primi ai expresiei logice ce descrie un circuit exista numerosi algoritmi, cel mai notabil fiind algoritmul Quine-McCluskey. In cazul proiectantului de circuite integrate este important sa se cunoasca restrictiile privind minimizarea logica,.

- Algoritmul Quine-Mc Cluskey optimizeaza numai functiile cu o singura iesire.
- Un circuit combinational reprezentat ca o implementarea a unei functii cu numar minim de implicanti primi nu furnizeaza in mod necesar si cel mai rapid circuit. Cand se are in vedere viteza, trebuie sa se considere si posibilitatile de comanda ale circuitului.
- Functiile minimizate nu conduc intotdeauna la cea mai mica arie pentru structura ca in cazul PLA-urilor.
- Circuitele minimizate pot fi mai dificil de testat sau mai putin tolerante la defecte. Pentru a creste fiabilitatea se poate introduce in mod intentionat redundanta.

7.3.4. Compilarea Ariilor de Porti.

Uneltele de proiectarea folosite la Ariile de Porti sunt similare cu cele utilizate pentru proiectarea circuitelor la cerere (ASIC). Compilarea ariilor de porti este in principal o operatie de rutatre, care va fi discutata in sectiunea urmatoare.

7.4. Amplasarea si rutarea.

Cele mai frecvent utilizate unelte de sinteza sunt *amplasarea* si *rutarea*. Acestea sunt folosite in special in cadrul metodelor de proiectare cu celule standard, unde specificatiile de proiectare sunt exprimate ca un set de porti conectate impreuna. Fiecare poarta corespunde, de regula, unei operatii logice combinationale sau unui sistem secvential de tipul unui bistabil. Planurile mastilor portilor sunt disponibile in cadrul unei biblioteci.

Uneltele de amplasare si rutare vor amplasa mai intai celulele in cadrul proiectului si apoi vor efectua conexiunile de rutare. Uneltele de amplasare si rutare trebuie sa functioneze impreuna, iar selectarea algoritmilor de amplasare si rutare nu se efectueaza independent, astfel, aceste unelte vor fi examinate simultan.

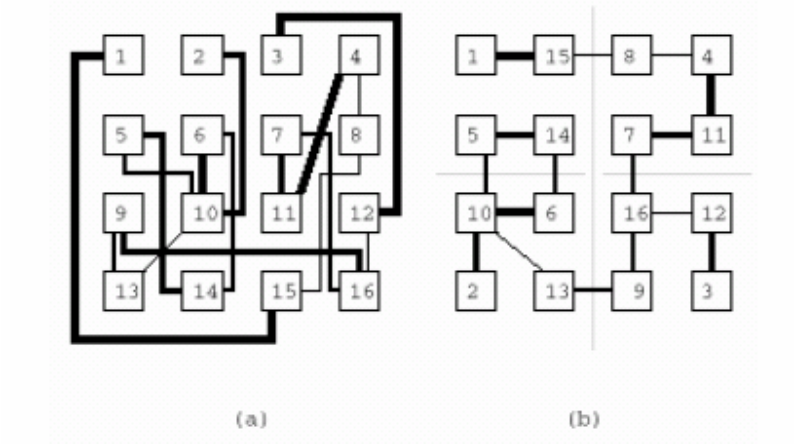
7.4.1. Algoritmi de amplasare.

Algoritmii de amplasare sunt cel mai dificil de realizat intr-o maniera eficienta. Scopul uneltelor de amplasare, denumite si “floor planners”, este acela de a amplasa blocurile proiectului astfel incat uneltele de rutare sa permita conectarea corespunzatoare a porturilor acestora. Aceasta problema este NP-completa (complexitatea creste aici exponential cu dimensiunea problemei). Ca rezultat nici unul dintre algoritmii utilizati in mod comun nu sunt optimali. Proiectantul, in nici un caz, nu este interesat de un proiect optimal, ci de un proiect care va satisface specificatiile de proiectare. In cazul unui plan de masti restrictiile se pot referi la o arie ocupata data si la satisfacerea anumitor limitari de timp.

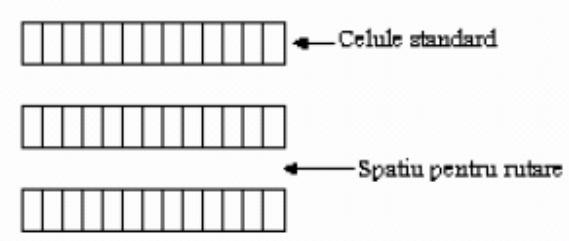
7.4.1.1. Algoritmul bazat pe taietura minima (min-cut).

Unul dintre cei mai simpli algoritmi folositi pentru amplasare este cel bazat pe *taietura minima*. Se determina conectivitatea proiectului iar circuitul este partitionat recursiv in jumatati egale. Fiecare partitie este astfel realizata incat un numar minim de conexiuni sa traverseze partitia. Aceasta tehnica de “divizare si cucerire” poate genera rezultate rezonabile. In figura de mai jos este prezentat un exemplu de algoritm bazat pe *taietura minima*. In figura (a) se prezinta blocurile proiectului. Numarul de conexiuni intre

blocuri este reprezentat de grosimea liniei de conectare. Figura (b) arata plasamentul final cu liniile de taiere. Mai intai este taiata linia orizontala, iar apoi cele doua linii verticale.
Un astfel de algoritm permite ca amplasarea proiectului sa puna accent pe reducerea lungimii totale a firelor. Aceasta conduce la un proiect de performanta ridicata, care ocupa o arie redusa.

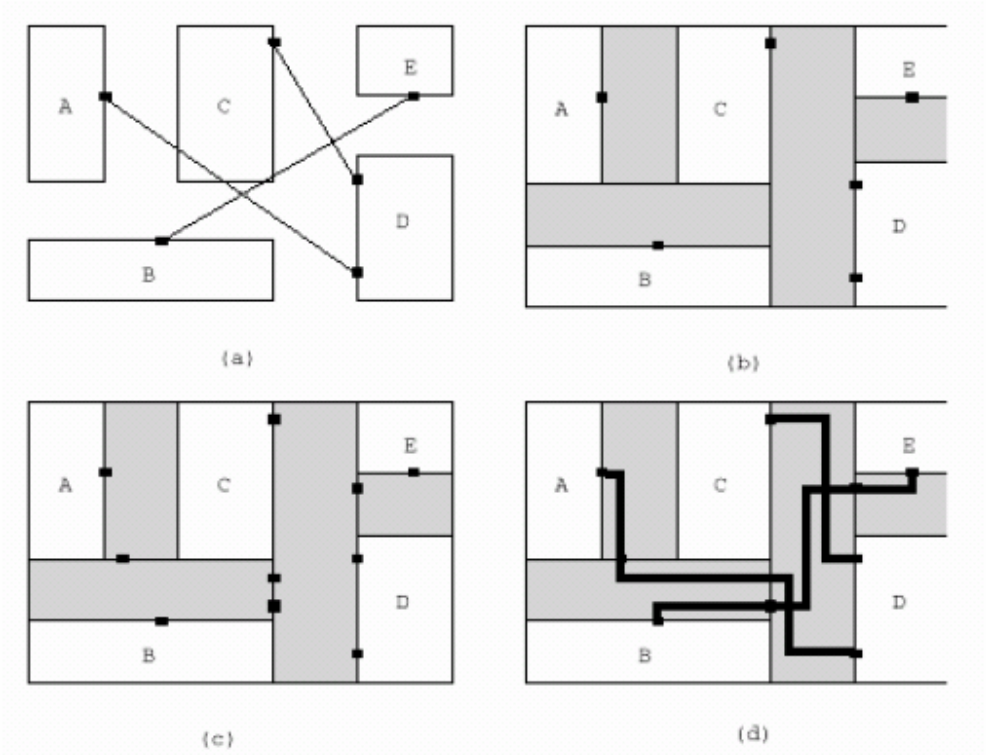


4.2.1.2. Amplasarea liniilor in cazul celulelor standard.
In cazul utilizarii metodologiei celulelor standard, amplasarea celulelor poate fi simplificata de cunoasterea structurii celulei. In mod tipic celulele standard sunt create cu trasee/sine de alimentare si masa la partile superioara si inferioara ale celulei. Celulele pot fi adiacente fara a conduce la violarea regulilor de proiectare (figura de mai jos). Amplasarea celulelor standard poate fi realizata prin amplasarea celulelor in linii, separate prin canale utilizate pentru rutare. Pentru a determina pozitia celulei in cadrul liniei trebuie folositi algoritmi bazati pe taietura minima sau pe alte tehnici. Liniile de celule mai contin si celule speciale, care permit conectivitatea intre liniile de celule.



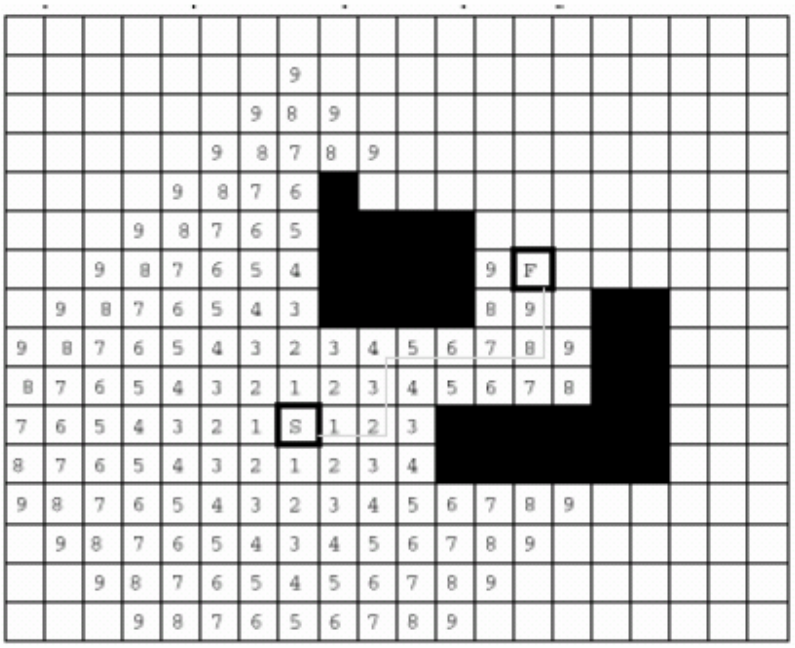
7.4.1.3. Calirea simulata.
Complexitatea algoritmilor de amplasare face ca algoritmul de calire simulata sa devina extrem de atractiv. Acesta va solicita insa mult timp de unitate centrala. Calirea simulata este algoritmul de baza care modeleaza calirea/cresterea rezistentei unui lichid sau solid. Calirea simulata este foarte atractiva si pentru faptul ca este intuitiva si usor de implementat.

7.4.2. Algoritmi de rutare.
Algoritmii de rutare sunt responsabili pentru conectarea porturilor in cadrul unui proiect in care a avut loc deja amplasarea.
Procesul de rutare este prezentat in figura de mai jos. Cele trei etape ale procesului de rutare sunt urmatoarele.



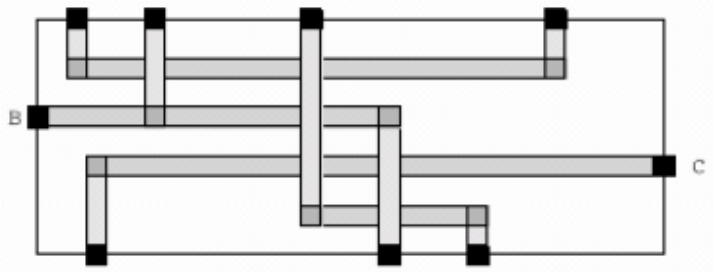
1. *Generarea canalului.* Se identifica regiunile prin care vor fi rutate/directionate conexiunile. In mod tipic canalele vor fi regiuni rectangulare. Regiunile nerectangulare vor fi partitionate in canale rectangulare (fig. b).
2. *Rutarea globala*. Daca trebuie rutata o retea intre doua canale, la frontiera intre cele doua canale se va crea un port. In aceasta etapa se determina canalele prin care se ruteaza o retea, dar nu se efectueaza conexiunile (fig.c).
3. *Rutarea canalului*. Sunt rutate canalele reale, adica sunt amplasate firele care efectueaza conexiunile (fig.d).
Rutarea reprezinta o procedura comuna in toate tipurile de ASIC-uri, pornind de la cele solicitate de catre utilizator, la celulele standard, la ariile de porti si la dispozitivele programabile de catre utilizator.
Ca si amplasarea, rutarea reprezinta o problema dificila. Intrucat cele mai multe rutoare nu ofera solutii optimale este important ca retelele sa fie tratate pe baza de prioritati. Astfel, retelele cu caracter critic vor fi rutate mai eficient. In continuare vor fi prezentati pe scurt unii algoritmi de rutare.

7.4.2.1. Algoritmul de rutare Lee-Moore.
Algoritmul Lee-Moore ruteaza cate un fir la un moment dat, fiecare conexiune efectuandu-se pe drumul cel mai scurt. Se realizeaza un tablou de celule ca in figura de mai jos.



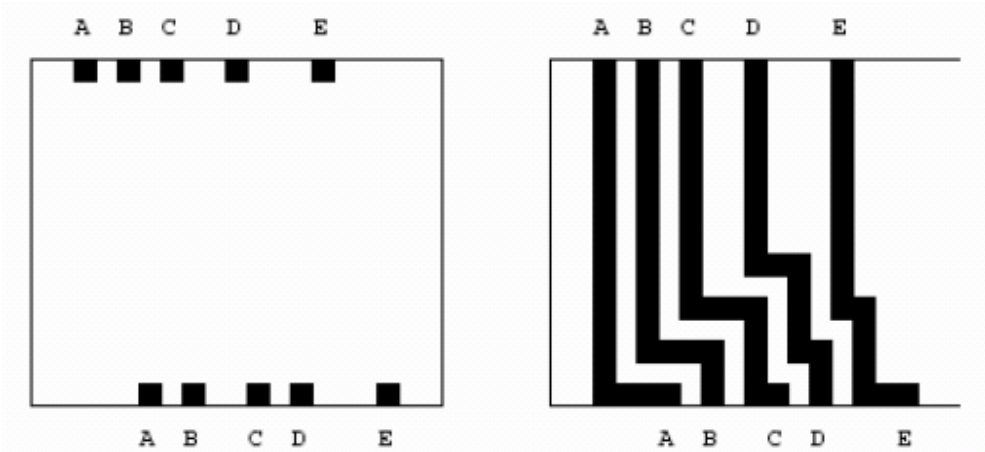
Punctele de plecare si oprire sunt notate cu S si F. Celulei de plecare din tablou *i* se atribuie valoarea *zero*. Celulelor adiacente acesteia li se atribuie valoarea *unu*. De fiecare data cand unei celule i se atribuie o valoare, valoarea incrementata se transfera tuturor celulelor vecine carora nu li s-a atribuit in ca o valoare.
Algoritmul se termina atunci cand se atribuie o valoare celulei finale. Oricare drum de la punctul final la punctul de start, care urmeaza o cale de numere in secventa reprezinta drumul cel mai scurt intre porturi. Dupa asignarea unui drum, se incearca conexiunea urmatoare.

7.4.2.2. Rutarea Canalelor.
Adesea este mai convenabil si eficient sa se ruteze simultan mai multe linii. Acesta este principiul pe care se bazeaza rutorul de canal. Un rutor de canal plaseaza fire parlele in directia principala a canalului.
In canal,liniile orizontale sunt rutate orizontal, iar liniile verticale sunt rutate vertical. In figura de mai jos se prezinta un canal rutat orizontal.



Fiecare retea are o linie orizontala, care conecteaza toate porturile in lungul canalului. Unul din dezavantajele rutarii canalelor consta in aceea ca un semnal fiind rutat prin canale multiple va trebui sa schimbe straturile frecvent. Astfel, procesul de fabricatie va trebui sa asigure mai multe straturi de metal.

7.4.2.3. Rutarea de tip rau (*river*).
In cazurile in care se stie ca firele nu se vor intersecta, se va putea folosi un algoritm de rutare de tip rau eficient. In figura de mai jos, porturile sunt conectate prin selectarea portului din stanga si rutarea in jos, iar apoi direct la dreapta, catre alt port.



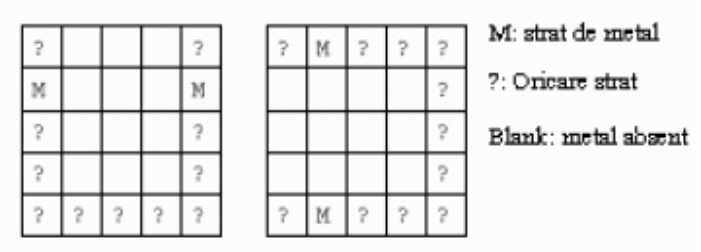
Urmatorul fir este inserat intr-o maniera similara, cu exceptia faptului ca va fi plasat in apropierea firului precedent la distanta minima de separare. Procedand in acelasi mod cu toate firele rutarea poate fiobtinuta la lungime minima . Aceasta tehnica nu se poate aplica in cazul in care firele se intersecteaza.

7.5. Tehnici de verificare.
In aceasta sectiune se vor discuta uneltele mai importante in procesul de verificare a proiectelor. Se vor avea in vedere *Verificatoarele de Reguli de Proiectare* (VRP), *Uneltele de Eextragere* si *Verificatoarele de Reguli Electrice*.

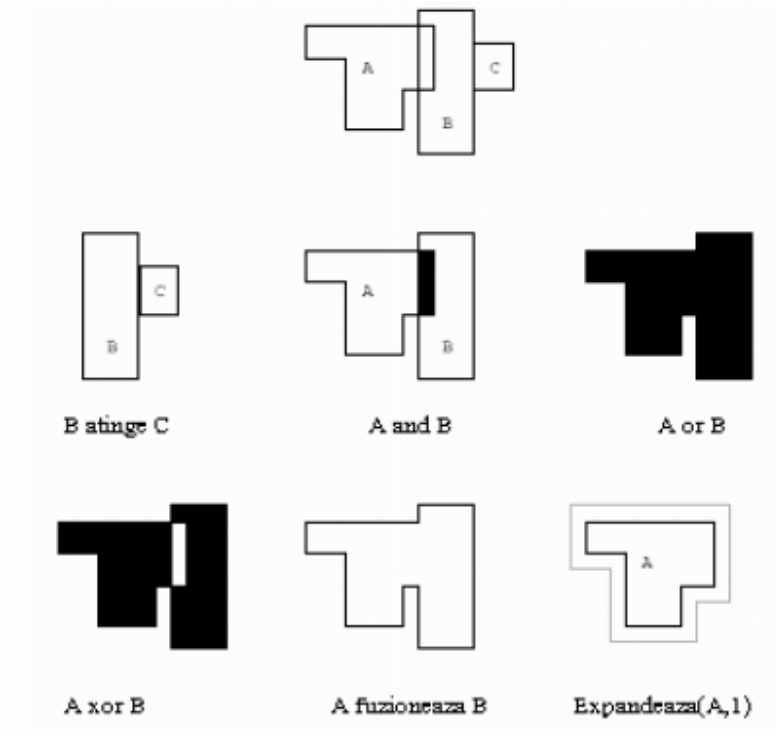
7.5.1. Uneltele VRP.
Verificarea Regulilor de Proiectare reprezinta un proces prin care se examineaza daca planul mastilor corespunde restrictiilor impuse de catre procesul de fabricatie. Regulile de proiectare sunt specificate ca un set de restrictii ce guverneaza separarile si conectivitatea in cadrul proiectului. O violare a regulilor de proiectare reprezinta o geometrie, care nu satisface regulile date.
In proiectele abordate ierarhic, VRP ierarhica joaca un rol important. Implementarea ierarhica a unui program VRP poate conduce la o importanta economie de timp. Tipic, aceasta implica verificarea completa a celulelor de pe cel mai coborat nivel si apoi efectuarea verificarii nivelului superior numai in ceea ce priveste periferia celulelor. Doua celule plasate alaturat trebuie sa fie verificate numai in ceea ce priveste cea mai mare distanta de la margine celulei.
Pentru ca un program VRP sa identifice corect violarea regulilor, el trebuie sa poata determina informatia de conectivitate. In plus, fata de regulile de proiectare, trebuie sa se mai dea un set de reguli de conectivitate. Astfel, un exemplu tipic este acela al regulilor privind straturile *metal*1 si *metal*2, care sunt conectate prin *via*. Cand se detecteaza *via* intre doua staturi se presupune ca ele trebuie sa fie conectate.

7.5.1.1. Tehnici de tip rastru.
Cea mai simpla tehnica, pentru determinarea violarilor de reguli simple de proiectare, se bazeaza pe utilizarea unei ferestre rastru, care trece peste planul mastilor pentru a stabili eventualele violari ale regulilor simple de proiectare. In figura de mai jos se prezinta ferestrele care trebuie trecute peste planul mastilor pentru a determina separarile straturilor de metal mai mici de 3 μm, cu toate

geometriile pe limite de 1 μm. Daca fereastra se potriveste cu forma de sub ea, s-a gasit o violare a regulilor de proiectare. Tehnica are avantajul ca este relativ simpla, dar ea nu poate sa foloseasac informatia de conectivitate globala si poate raporta violari inexistente. Daca doua linii de metal conectate sunt separate la mai putin de 3 μm se raporteaza o violarea a regulilor de proiectare. Geometriile non-mahattan nu pot fi verificate corect cu aceasta tehnica.



7.5.1.2. Tehnici bazate pe algebra poligoanelor.
Tehnicile bazate pe algebra poligoanelor folosesc sase operatii: *and, or, xor, atinge, fuzioneaza, expandeaza*. Cele sase operatii sunt prezentate in figura urmatoare.



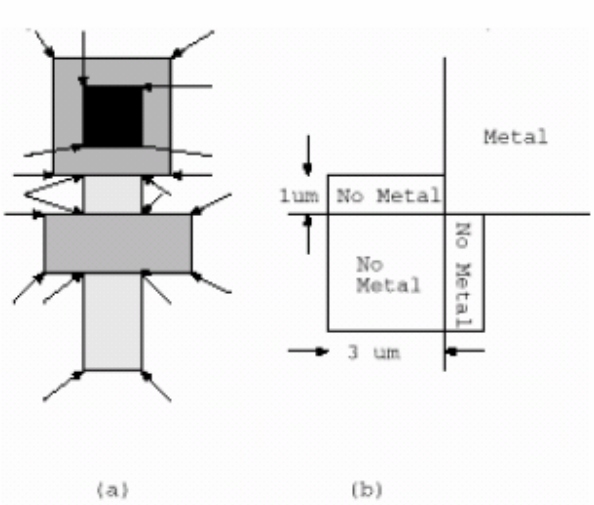
O regula de proiectarea, pentru o separare intre traseele de metal de 3 μm, se poate exprima astfel:

EROARE = expandeaza (poligon, 1.5μm) and expandeaza(poligon2, 1.5μm)

Aceste tehnici sunt computational intensive.

7.5.2. Tehnici bazate pe varfuri.
Ultima tehnica VRP utilizeaza varfurile din proiect. Varful este definit a fi locul in care se intalnesc doua muchii. Varfurile unui plan pentru masti sunt aratate in figura de mai jos (a). Fiecare regula de

VRP are un sablon asociat pe care il verifica la fiecare varf. Sablonul pentru spatierea de metal de 3 in (b).



Aceasta tehnica este in general aplicabila geometriilor Manhattan, dar paote fi extinsa si la geometriile non-Manhattan.

7.5.3. Unelte de Extractie.

Dupa generarea planului mastilor, este important sa se extraga din acesta parametri electrici. In mod special intereseaza identificarea urmatoarelor elemente ale planului: tranzistoare, contacte, capacitori si rezistente. Algoritmii utilizati pentru identificarea elementelor sunt foarte asemanatoari cu cei dintr-un program VRP. De fapt cele mai multe unelte de extractie implementeaza si functiile de VRP. Elementele de circuite CMOS sunt identificate dupa cum urmeaza:

- Tranzistoarele apar acolo unde un traseu de siliciu policristalin intersecteaza un traseu de difuzie. Trebuie sa se calculeze aria tranzistorului, latimea si lungimea trebuie calculate. Conexiunea la substrat trebuie inregistrata.
- Capacitatile apar intre toate straturile. Atunci cand un strat intersecteaza alt strat, pentru calcularea capacitatii se va folosi aria de intersectie. Aceasta implica calcularea ariilor si a perimetrelor pentru straturile poligonale.
- Rezistorii apar pe toate liniile. Rezistenta depinde de lungimea si grosimea liniei. Trebuie avute in vedere si trseele curbate. Rezistenta este exprimata in unitati: Ω/patrat.

7.5.4. Verificatoare de Reguli de Natura Electrica (VRNE).

VRNE sunt programe care examineaza informatia din lista componentelor/conexiunilor extrasa din planul mastilor si verifica incalcarea diverselor reguli de natura electrica. In continuare se prezinta cateva dintre regulile verificate:

- *Scurt circuite*: Retelele care sunt conectate la masa sau la tensiunea de alimentare sunt semnalate ca fiind erori posibile.
- *Circuite izolate*: Retelele care nu sunt comandate de nici un semnal se considera a fi eronate.
- *Puterea consumata*: este posibila estimarea puterii consumate si verificarea curentilor in trasee.
- *Rapoartele tranzistoarelor*: Exista posibilitatea de a verifica daca rapoartele tranzistoarelor din circuit sunt rezonabile. Ca erori pot fi semnalate nepotrivirile intre dimensiunile tranzistoarelor NMOS si PMOS, in circuitele CMOS, tranzistoarele de dimensiuni mici, care comanda sarcini mari sau porti OR cablate.
- *Compararea retelelor*: Lista conexiunilor poate fi comparata cu schema originala a circuitului pentru a verifica daca a fost generat acelasi circuit. Acest lucru este dificil daca specificarea

circuitului original a fost structurala si daca pe parcursul sintezei planului mastilor au fost efectuate optimizari. De exemplu, daca au fost inlaturate portile redundante.

7.5.5. Verificarea.
Verificarea la nivel temporal. Circuitul este analizat pentru a gasi cazul cel mai defavorabil privind intarzierile. Daca acestea nu satisfac specificatiile circuitului se semnaleaza o eroare.
Verificarea functionala. Circuitul este analizat pentru a gasi portile logice de baza. Comportarea circuitului este dedusa din acestea. Se poate efectua, in continuare, compararea cu specificatiile de comportare ale circuitului original..
Amplasarea si rutarea bazate pe criterii de performanta se utilizeaza in cadrul uneltelor de proiectare a circuitelor integrate moderne. Daca se detecteaza o violare a criteriului de timp, in cadrul procesului de amplasare si rutare, se poate folosi o alta strategie de implementare. De exemplu, daca un inmultitor nu satisface criteriul de performanta se poate utiliza intr-o masura mai mare tehnica bazata pe banda de asamblare. Aceasta va conduce la o arie mai mare dar si la performante potentiale mai mari.

7.6. Unelte de simulare.
Simularea proiectelor reprezinta o etapa importanta in ciclul de proiectare a circuitelor integrate. Scopul simularii este acela de a determina comportarea proiectului inaintea fabricarii acestuia. In aceasta sectiune se vor examina simulatoarele folosite in proiectarea circuitelor integrate.

7.6.1. Simulatoare la nivelul circuitului.
Simulatoarele la nivelul circuitului sunt utilizate pentru a determina caracteristicile circuitului ca forme de unda. Uneltele de extractie sunt folosite pentru a produce lista conexiunilor ce contine componentele electrice ale circuitului, inclusiv capacitatile parazite si rezistentele. Simulatoarele preiau aceasta informatie si genereaza un sistem de ecuatii diferentiale neliniare. Solutia acestui sistem ofera analiza comportarii in regim tranzitoriu a circuitului.

7.6.1.1. Spice.
Cel mai raspandit program de simulare la nivel de circuit este SPICE. Initial acest program a fost dezvoltat in anii 70, la Berkeley, si a fost scris in FORTRAN si C. Codul sursa pentru SPICE este disponibil si a fost portat pe aproape toate tipurile de calculatoare. Sunt disponibile, de asemenea, si versiuni comerciale: HSPICE, PSPICE. In plus cei mai multi furnizori de circuite integrate ofera componentele si lista lor de conexiuni pentru proiectele lor.
SPICE efectueaza analiza in CC prin tratarea capacitorilor ca circuite deschise, iar a inductorilor ca scurtcircuite.
Pentru noul circuit este creata o matrice, care este solutionata. Intrucat ecuatiile sunt neliniare, se foloseste o tehnica iterativa cum ar fi metoda Newton-Raphson. Elementele de circuit sunt reprezentate in matrice sub forma:

$Y_b V_b + Z_b I_b = W_b$

unde:
- V_b este matricea tensiunilor pe ramuri;
- I_b este matricea curentilor pe ramuri;
- W_b este matricea conditiilor initiale.

De exemplu: un rezistor are $Y_b = 1$, $Z_b = -R_b$ si $W_b = 0$. Aceasta va furniza $V = IR$.
Un capacitor are $Y_b = sC_b$, $Z_b = -1$ si $W_b = C_b V_0$. Aceasta va genera ecuatia capacitorului ca

$sC_bV_b - I_b = C_bV_0$.

Se stie din prima teorema a lui Kirchoff:

$AI_b = 0$
$V_b - A'V_n = 0$

unde:

- A este matricea de incidenta $a_{i,j} \in (-1, 0, 1)$

Matricea utilizata pentru a reprezenta elementele de circuit, adesea denumita tablou, este:

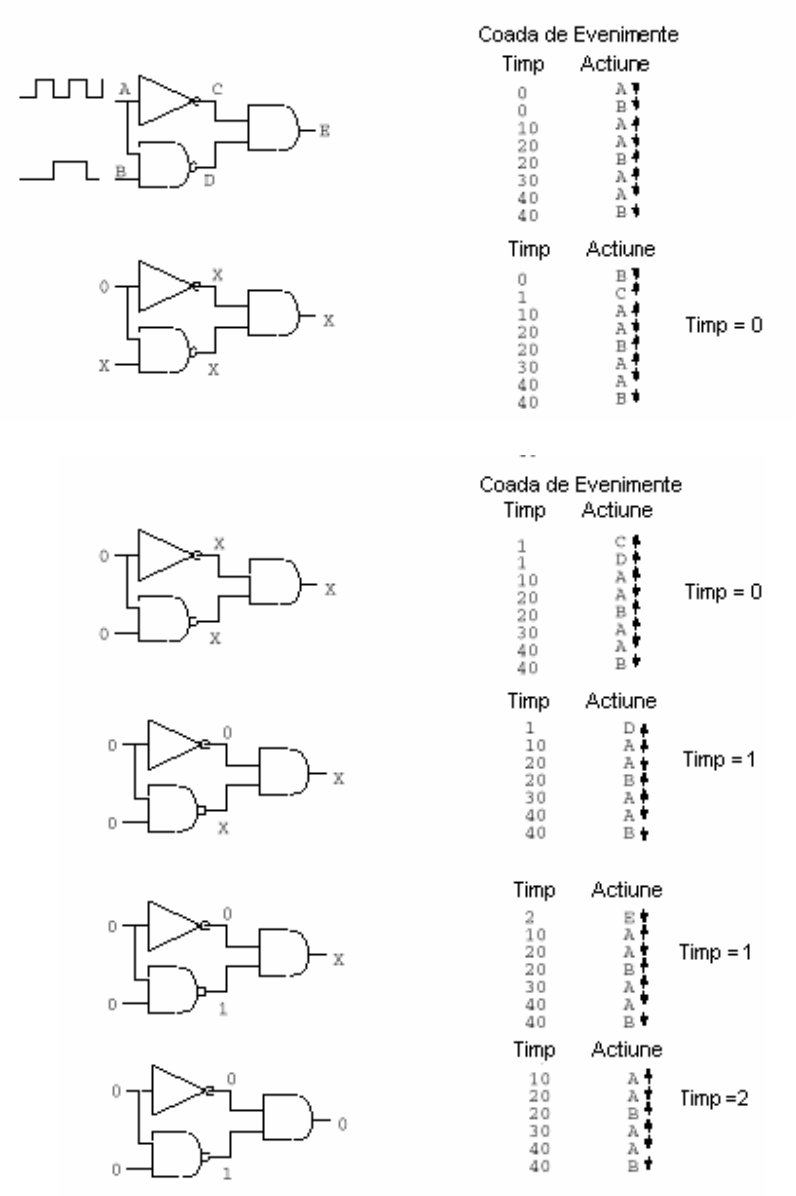
$$\begin{bmatrix} 1 & 0 & -A^t \\ Y_b & Z_b & 0 \\ 0 & A & 0 \end{bmatrix} \cdot \begin{bmatrix} V_b \\ I_b \\ V_n \end{bmatrix} = \begin{bmatrix} 0 \\ W_b \\ 0 \end{bmatrix}$$

Acest tablou, cand este solutionat, va determin tensiunile la toate nodurile, cat si tensiunile si curentii dea lungul tuturor ramurilor.
Analiza procesului tranzitoriu este efectuata dupa cum urmeaza:
- Daca nu sunt specificate conditiile initiale se efectueaza analiza in CC.
- Se sparge intervalul de simulare in pasi de timp discret.
- Se integreaza numeric ecuatiile diferentiale in ecuatii algebrice echivalente. Pentru aceasta se poate folosi Metoda Euler inversa.
- Se insereaza aceste noi ecuatii in matrice, care se solutioneaza prin tehnici de analiza de CC.
- Se repeat pana cand simularea se termina.

Alte caracteristici ale pachetului nSPICE sunt:
- SPICE suporta analiza de CC. Analiza in regim tranzitoriu si analiza in CA, pentru semnale mici.
- SPICE suporta tehnologii diferite: MOS, Bipolar etc.
- Pentru circuite mari complexitatea analizei o face nerealista. Tipic sunt simulate portiuni ale proiectului.
- Uneori SPICE nu poate ajunge la o solutie convergenta, iar pentru simulare trebuie efectuate ajustari ale parametrilor.
- SPICE suporta modele diferite pentru tranzistoarele MOS.

7.6.2. Simulatoare Comandate de Evenimente.

In cazul in care intereseaza numai comportarea numerica a circuitului, nu mai este eficienta rezolvarea intregului circuit pentru a afisa starile, de fiecare data cand un nod isi schimba tensiunea. O alternativa se bazeaza pe simularea comandata de evenimente, in care numai partea de circuit afectata de schimbarea de stare este simulata.
Fie circuitul din figura de mai jos. Sunt prezentate starile initiale. Se urmareste simularea circuitului avand in vedere impulsurile aplicate la intrarea portii. Fiecare poarta are o intarziere de 1 ns, iar semnalele de intrare se modifica la intervale de 10 ns. Coada de evenimente pastreaza o lista de evenimente, care urmeaza sa apara la momente specificate de timp. Simularea circuitului, plecand de la continutul cozii de evenimente, este prezentata pentru primele 2 ns. Sunt necesare numai cinci evaluari ale portii (trei care au provocat evenimente ce au fost planificate si doua care nu au fost planificate). Pentru ca circuitul intreg sa fie recalculat pentru fiecare nod, sunt necesare noua evaluari ale portii.



7.6.3. Simulare la nivel de comutator.

In timp ce simularea circuitului ofera rezultate corecte la nivelul formelor de unda, adesea trebuie testate numai functionalitatea circuitului. Simulatoarele la nivel de comutator modeleaza tranzistoarele ca simple comutatoare, care transmit sau nu o valoare. Nivelurile de tensiune sunt sunt modelate prin una dintre cele trei stari: 0-logic, 1-logic sau x-nedeterminat. Cu fiecare stare se asociaza puterea semnalului. Starile si puterile pentru un simulator tipic sunt aratate in laticea din figura (a) de mai jos. Puterile au urmatoarele semnificatii:

D *Comandat*. Aceasta este puterea atribuita lui V_{DD}, V_{SS} si semnalelor de intrare.

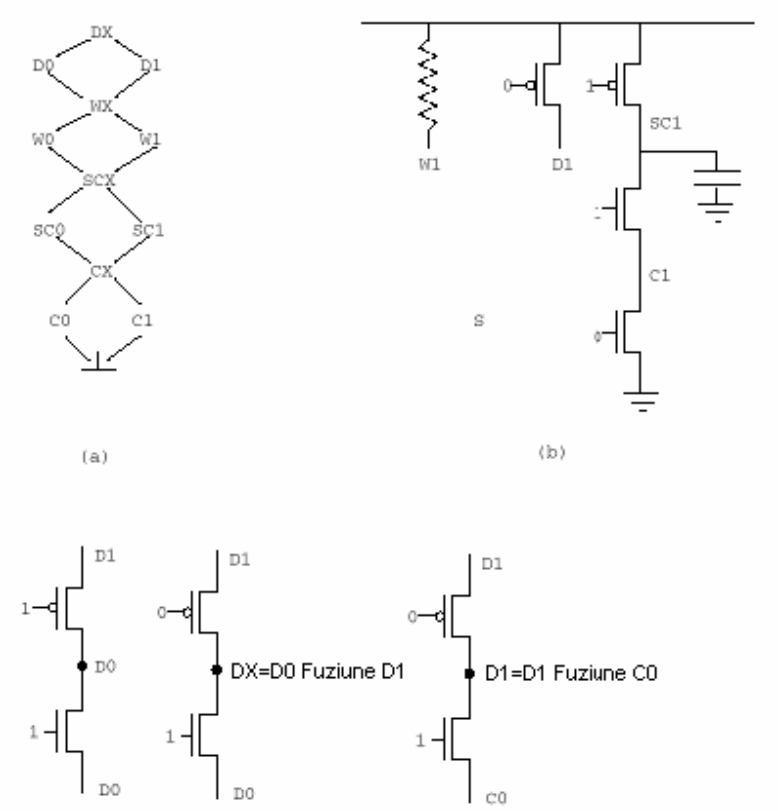
Comandat presupune ca nodul este comandat de catre sursa de curent.

W *Comandat slab*. Acesta este ca si cu ar fi comandat, dar comandat relativ slab.

Un tranzistor trage-sus NMOS va comanda un nod catre o stare comnadata slab.

SC *Supraincarcat*. Aceasta este puterea unui nod a carui valoare este stocata pe un capacitor, iar starea a fost asignata de puterea de comanda.

C *Incarcat*. Aceasta este puterea unui nod a carui valoare este stocata pe un capacitor, dar a carui putere nu a fost stabilita de catre un semnal de comanda.
⊥ *Nivelul cel mai coborat.*
Figura (b) prezinta cateva circuite care ilustreaza cazuri in care pot fi intalnite diferite puteri.



In functie de precizia ceruta, simulatoarele pot folosi aceste stari si puteri sau un superset de subseturi ale acestora.
Starea si puterea ale unui nod sunt determinate prin gasirea celui mai coborat punct din latice, care are o cale in jos catre toate starile pe care le comanda. Fie circuitele din figura de mai sus. Primul circuit arata un inversor. Numai unul dintre tranzistoare conduce (NMOS) astfel iesirea capata starea D0. Urmatorul circuit are in conductie ambele tranzistoare NMOS si PMOS, astfel ca iesirea tinde spre D1 Fuziune D0 = DX, o valoare nedeterminata. Cel de-al treilea circuit are iesirea fortata la D1, intrucat D1 este mai puternic decat nodul incarcat C0.
Iata cateva dintre simulatoarele la nivel de comutator:
- SILOS reprezinta un simulator commercial care suprta 12 niveluri logice (Alimentare, Comanda, Rezistiva si Impedanta ridicata, pentru fiecare 0, 1 si X). SILOS permite simularea la nivel de tranzistoare si la nivel de porti. Pot fi utilizate, de asemenea, modele comportamentale.
- VHDL descrie circuitele intr-o maniera convenabila pentru simularea bazata pe evenimente discrete. Tipul construit in program BIT defineste numai nivelurile logice 1 si 0, dar pot fi utilizate niveluri diferite. Functia Fuziune poate fi descrisa de functiile de rezolutie privind magistrala.

7.6.4. Simulator la nivel de porti.
In locul modelarii circuitelor la nivelul tranzistoarelor, simularea poate fi efectuata la nivel de poarta.

Simularea se poate realiza plecand de la schema cu componentele la nivel de poarta si de la un simulator cu modele pentru fiecare tipde poarta. Aceasta permite captarea si simularea proiectelor independent de tehnologia de implementare. Functionalitatea portilor trebuie sa fie aceeasi independenta de procesul de fabricatie. Dupa terminarea proiectului, se poate selecta tehnologia si efectua simularea la nivel de comutator sau de element de circuit.
- SILOS asigura modele la nivel de porti pentru functiile logice commune, latch-uri si bistabile.
- VHDL este potrivit pentru modelarea circuitelor la nivel de porti.

7.7. VHDL.

VHDL sau VHSIC HDL (Very High Speed Integrated Circuits Hardware Description Language) a fost dezvoltat in cadrul Programului VHSIC al Departamentului Apararii al SUA. Limbajul a fost ulterior modificat si acceptat ca standard IEEE, in 1987. In aceasta sectiune se va prezenta pe scurt VHDL. In mod curent se pot intalni si alte limbaje de descriere a hardware-lui (HDL), VHDL fiind selectat aici deoarece este des intalnit in proiectarea circuitelor integrate. Iata cateva dintre caracteristicile limbajului.
- VHDL reprezinta un limbaj HDL de nivel inalt, care asigura descrieri de la nivel de poarta la nivel de sistem. Nu sunt suportate modele de tranzistoare sau de masti.
- VHDL este asemanator limbajului de programare ADA. Sunt incluse declaratii de tip (scalar, masiv, inregistrare), chemari de subrutine, cicluri *if* si *do*, pointeri etc.
- VHDL opereaza si cu tipul fizic pentru a asigura informatia de sincronizare/timing.
- VHDL face distinctie intre semnale si variabile. Semnalele sunt utilizate pentru a reprezenta retelele de circuite, in timp ce variabilele sunt folosite pentru a pastra informatia de stare a programului.
- VHDL suporta derscrieri generice parametrizate. De exemplu: sumatorul pe N biti.
- VHDL suporta descrieri Structurale. Acestea sunt descrieri de proiect, care prezinta proiectul sub forma de componente si conectivitate.
- VHDL suporta descrieri de tip Flux de Date (Dataflow), Descrierea de tip Flux de date prezinta fluxul datelor prin sistemul proiectat, similar cu Limbajul Transferurilor intre Registra (RTL – Register Transfer Language).
- VHDL suporta descrieri Comportamentale. Acestea sunt utilizate pentru a surprinde comportarea dinamica a circuitelor.

Descrierile de proiecte in VHDL constau in urmatoarele unitati de proiectare.
- *Entitatea Interfata* (Entity Interface) descrie conexiunile externe ale proiectului.
- *Entitatea Corp* (Entity Body) descrie implementarea sau comportarea unei entitati.

Fiecare interfata entitate are una sau mai multe entitati de tip corp, asociate cu ea.
- *Configuratie* (Configuration). Aceasta selecteaza care entitate corp va fi folosita in constructia unei ierarhii a proiectului.
- *Pachet* (Package) contine declaratii commune si specificatii, care sunt partajate cu alte unitati de proiectare.
- *Corp Pachet*(Package Body) contine subprogramele de clarate in Pachete.

7.7.1. Exemplu de Descriere Structurala.

```
--
--       example VHDL program for simple
--       arithmetic unit
--
package misc is
   type control_struct is record
      con0   : bit;
      con1   : bit;
   end record;
end misc;
use misc;
entity testx6 is
   generic
      (
      N : natural := 3
      );
   port
      (
      A : bit_vector(0 To N);
      B : bit_vector(0 To N);
      Output : bit_vector(0 to N);
      Con : control_struct;
      Cin : bit;
      Cout : bit
      );
end testx6;
use misc;
architecture atestx6 of testx6 is
   signal B_not    : bit_vector(0 To N);
   signal X1,X2,X3 : bit_vector(0 To N);
   signal Carry    : bit_vector(0 To N-1);
begin
   component inv port (
     input : inout Bit;
     output : out Bit);
   component nand2 port(
      Ain     : in Bit;
      Bin     : in Bit;
      Output  : out Bit);
```

```
component fadd port(
   Ain,Bin,Cin : in Bit
   Sum,Cout    : out Bit);
for i in 0 To N generate
   inv port map(B(i),B_not(i));
   nand2 port map(B(i),Con.con0,X1(i));
   nand2 port map(B_not(i),Con.con1,X2(i));
   nand2 port map(X1(i),X2(i),X3(i));


      if i = 0 generate
        fadd port map(a(0),X3(0),
  Cin,Output(0),Carry(0));
      end generate;
      if (i > 0) and (i < N) generate
        fadd port map(a(i),X3(i),Carry(i-1),
  Output(i),Carry(i));
      end generate;
      if i = N generate
        fadd port map(a(N),X3(N),
  Carry(N-1),Output(N),Cout);
      end generate;
   end generate;
end atestx6;
```

7.2.2.Exemplu de Descriere Comportamentala.

```
--
--       Full Adder behavioral description
--
entity fadd1 is
   port(Ain : bit := '0';
Bin : bit := '0';
        Cin : bit := '0';
Sum : bit;
Cout : bit);
end fadd1;
architecture fadd of fadd1 is
begin
   Sum <= Ain xor Bin xor Cin after 5ns;
   Cout <= (Ain and Bin) or (Ain and Cin)
      or (Bin and Cin) after 6ns;
end fadd;
```

7.8. Formate pentru Interschimb de Proiecte.

HDL-urile de tipul VHDL, discutat mai sus, sunt utilizate de catre proiectanti pentru a descrie circuitul, si astfel, pentru "captarea" proiectului. Formatele pentru Interschimb de proiecte sunt, de asemenea, limbaje pentru descrierea informatiei de proiectare, permitand interschimbul acesteia intre diferite programe. In mod curent aceste formate nu sunt proiectate pentru a fi citite sau scrise usor, ci pentru a descrie informatia intr-o maniera usor de analizat si generat. In industria circuitelor integrate se intalnesc formate de interschimb. In continuare vor fi trecute in revista aceste trei formate, cu unele exemplificari.

7.8.1. CIF.

- Formatul Intermediar Caltech (Caltech Intermediate Format).
- Comanda Layer specifica stratul current.
- Comanda BOX deseneaza dreptunghiuri. Comanda WIRE deseneaza fire. Comanda POLYGON deseneaza poligoane si comanda Roundflash deseneaza cercuri.
- Comanda CALL deseneaza o subrutina. DS indica inceputul unei subrutine, iar DD specifica sfarsitul unei subrutine. O subrutina este utilizata pentru a defini o celula (colectie de forme) intr-o ierarhie.
- Toate dimensiunile sunt date in sutimi de microni (10^{-8} m).
- Informatia specificata de catre utilizator poate fi stocata in subrutinele 0 – 9. Semantica acestora nu a fost standardizata.

Mai jos se prezinta o descriere CIF.

```
DS 1;
9 PROBE_PAD;
L CV;
P 32300,-14800 32800,-14800 32800,-14300 32300,-14300;
P -33700,-14800 -33200,-14800 -33200,-14300 -33700,-14300;
L CM2;
P 8500,3500 -9000,3500 -9000,-14000 -500,
-22500 -500,-34500 0,-34500 0,-22500
8500,-14000;
P -34000,-15100 -16500,-15100 -16500,3500 -34000,3500;
L CM;
P 33500,-14000 16000,-14000 8500,-21500 -9000,-21500
-16500,-14000 -34000,-14000 -34000,-31500 -9000,-31500
-9000,-34500 8500,-34500 8500,-31500 33500,-31500;
DF;
DS 2;
9 TLINE3;
C 1 R 0,1 T -136750,38900;
C 1 MY R 0,1 T 400250,38900;
L CM2;
P -102250,38400 365750,38400 365750,38900 -102250,38900;
L CM;
P -102250,29900 365750,29900 365750,47400 -102250,47400;
DF;
DS 29;
9 GREG_CHIP;
C 2 R 0,1 T -45150,70750;
DF;
C 29;
E
```

7.8.2. GDS II.

- GDS II a fost primul format pentru interschimb si este unul dintre cele mai raspandite in PAC a circuitelor integrate. El a fost dezvoltat de catre Calma Co. (General Electric)
- GDS II suporta text, masive, care nu sunt intalnite in CIF.
- Spre deosebire de CIF, GDS II este un format binar.

7.8.3. EDIF.

- EDIF – Electronic Design Interchange Format este utilizat in mod frecvent ca format pentru descrierea listei de componente/conexiuni necesara diferitelor sisteme de proiectare.
- EDIF este, din punct de vedere sintactic, similar cu LISP. Ambele sunt bazate pe liste.
- EDIF are trei niveluri, nivelul cel mai coborat nu este programabil si necesita constante in toate expresiile. Nivelul 1 suporta variabile si expresii, ca si celule parametrizate. Nivelul 2 este programabil si reprezinta o extensie a limbajului LISP.
- EDIF este deja acceptat ca standard.
- EDIF suporta diferite abordari ale proiectarii. Componentele si lista lor de conexiuni, Scheme, Comportament, Planul mastilor s.a.

7.8.3.1. Exemplu EDIF.

```
(
EDIF VHDL_design (EDIFversion 2 0 0)
   (EDIFlevel 0) (keywordmap (keywordLevel 0 ) )
  (status ( written (timeStamp 1990 24 7 19 30 27) ) )
  ( EXTERNAL SC_LIB (EDIFlevel 0) ( technology(
  (numberDefinition ( scale 1 (e 1 -9) (unit Distance))))
    ( cell and3 ( cellType GENERIC )
      ( view abstract (viewType NETLIST)
        ( interface
          ( port ( Rename GND "gnd!" ) (direction input) )
          ( port ( Rename VDD "vdd!" ) (direction input) )
          ( port ( Rename OUTPUT "Out" ) (direction output) )
  ( port ( Rename CIN "Cin" ) (direction input) )
  ( port ( Rename BIN "Bin" ) (direction input) )
  ( port ( Rename AIN "Ain" ) (direction input) )
        )
      )
    (COMMENT .... other cells delete from this description .... )
    ))
  (library (rename DEFAULT_LIB ".") (EDIFlevel 0)
  (technology (numberDefinition (scale 1 (e 1 -9)(unit Distance))))
    (cell NBIT_ALU1  ( cellType GENERIC )
      (view autoLayout (viewType NETLIST)
        (interface)
        (contents ( instance AND377 (
            viewref abstract
              (cellref and3 ( libraryRef SC_LIB ) ) ) )
          (instance AND3  ( viewref abstract (cellref and3
     ( libraryRef SC_LIB) )))))
    (COMMENT ....... other cell instances deleted .....)
    (net ( Rename VDD "vdd!")
      ( Joined (
      ( portref VDD ( instanceref INPAD85 ) )
      ( portref VDD ( instanceref INPAD84 ) )
      ( COMMENT ... rest of VDD connects deleted ..)
    ( net B_2_
     ( Joined (
          ( portref INPUT ( instanceref INPAD66 ) )
          ( portref AIN  ( instanceref AND248 ) )
          (Property sigType (string "signal") (owner "Cadence" ))
       )
       (COMMENT rest of nets deleted ...)
) ) ) ) )
```

7.9. Sinteza comportamentala.

Scopul uneltelor de sinteza comportamentala consta in preluarea unei specificari a comportarii proiectului pentru a genera implementarea structurala. Este ca si cum “tu spune ce trebuie sa faca circuitul iar uneltele realizeaza circuitul”. In aceasta sectiune se vor discuta elementele de baza pentru transformarea descrierilor comportamentale in descrieri structurale. Sinteza descrierilor structurale a fost deja examinata.

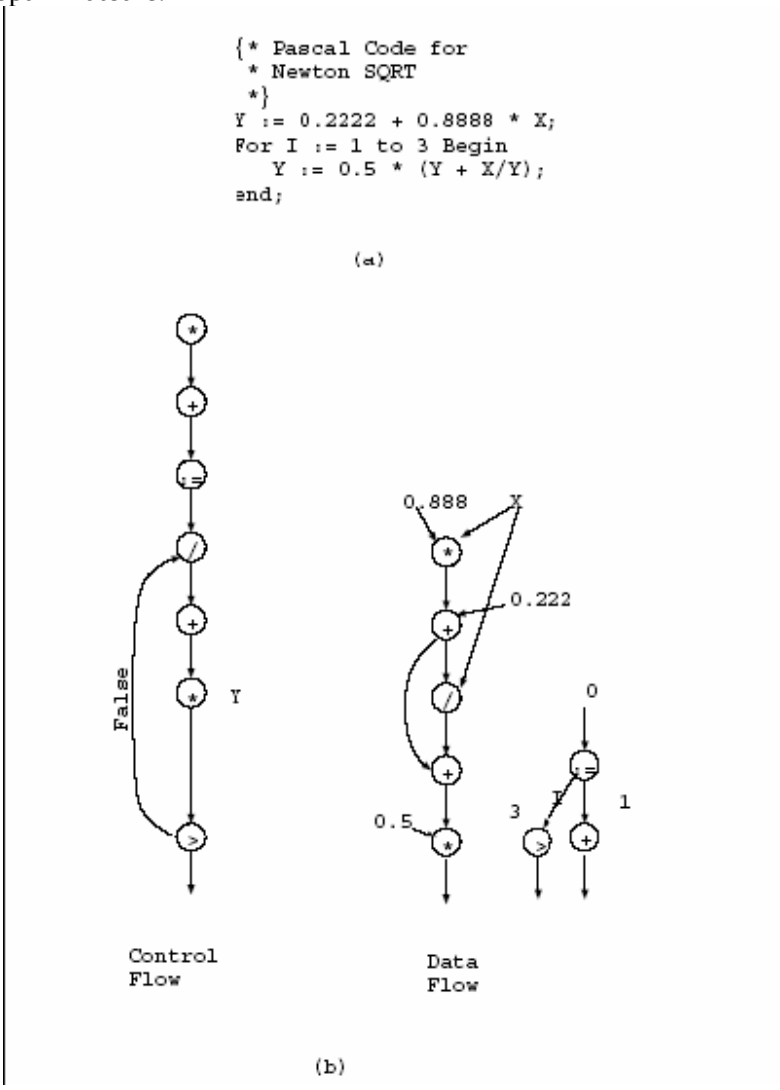
Prima motivatie pentru sinteza comportamentala este aceea de a reduce timpul de proiectare si a erorilor.

Sinteza permite, de asemenea, proiectantului sa exploreze rapid numeroase solutii pentru proiect, cat si metodologii diferite. Documentatia poate fi incorporata usor in descrierile de nivel inalt. Procesul de sinteza comportamentala poate fi structurat in trei etape.

7.9.1. Definirea Task-ului.

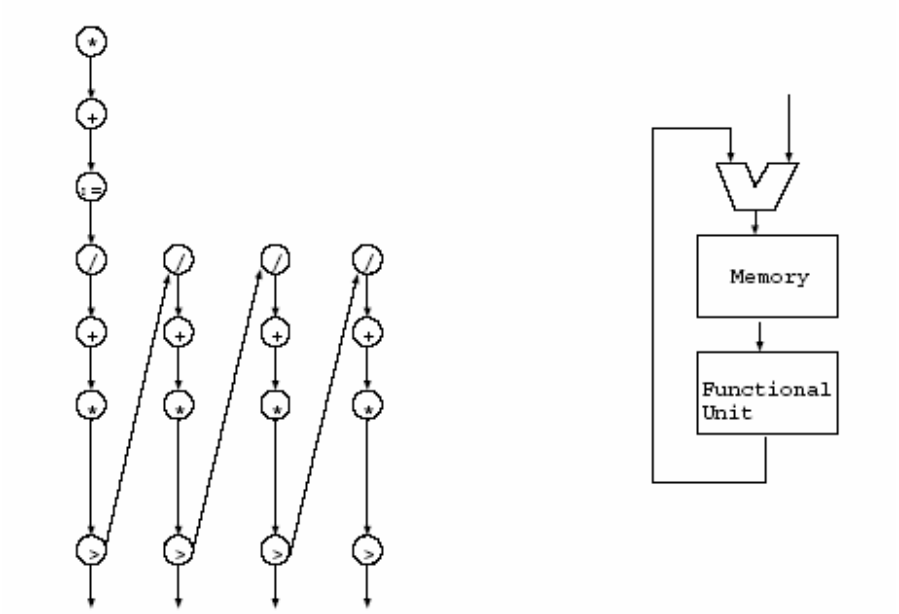
Etapa de definire a task-ului consta in citirea informatiei de intrare referitoare la descrierea comportamentala pentru a genera o reprezentare interna a proiectului. Sintaxa de intrare pentru descrierea comportamentala poate fi un limbaj de programare, de exemplu C, un HDL cum ar fi VHDL, Verilog sau un limbaj functional de programare ca Prolog sau Lisp. Descrierea comportamentala este apoi convertita

in reprezentari grafice, dintre care una contine informatia privind fluxul de control, iar cealalta informatia legata de fluxul de date. Un exemplu este dat in figura urmatoare, care se refera la calculul radacinii patrate a unui numar folosind metoda lui Newton. Algoritmul este descries in limbajul Pascal. Pe acest graf pot fi efectuate optimizari pentru eficientizare. Una dintre optimizari consta in substituirea inmultirii cu 0,5 printr-o deplasare. Algoritmii de optimizare sunt similari cu cei folositi de compilatoarle optimizatoare.

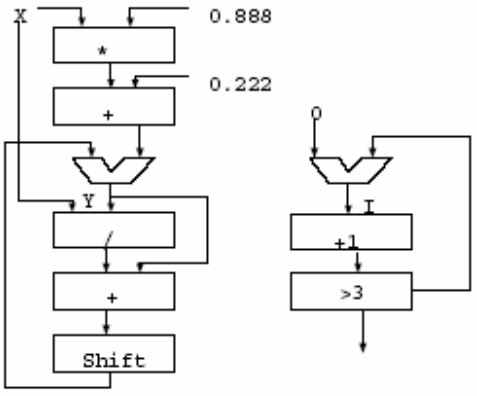


7.9.2. Planificarea.

In etapa de planificare se ia decizia privitoare la momentul in care trebuie sa aibe loc operatiile din graful de comanda. Planificarea pasilor in timp (pasii de comanda) in care trebuie efectuate operatiile. In figura de mai jos se prezinta o alocare posibila a operatiilor.



(a)



(b)

7.9.3. Alocarea.

Alocarea constituie un process de asignare a hardware-lui pentru operatiile date. Cel mai simplu algoritm atribuie un element separate de hardware fiecarei operatii. Hardware-ul poate fi minimizat prin partajarea unitatilor functionale, care efectueaza aceeasi functie in pasi diferit. In cadrul algoritmului de alocare pot fi introduse si alte restrictii.

7.10. Noi unelte de proiectare.

Pe masura ce proiectele de circuite integrate devin mai mari si mai complexe se impune realizarea de noi unelte pentru PAC. In continuare se dau cateva directii de cercetare.

- Uneltele de sinteza bazate pe descrierea comportamentala la nivel inalt prezinta in continuare interes. Nu exista unelte comerciale, care sa efectueze unele etape ale sintezei pornind de la descrieri comportamentale de nivel inalt.

- Unelte pentru proiectarea de siteme analogice. Uneltele descrise mai sus au avut in vedere proiectarea sistemelor numerice. In prezent se dezvolta pachete de proiectare pentru sisteme analogice plecand de la HDL-uri corespunzatoare.
- Incorporarea unor tehnici de testare in cadrul sintezei circuitelor. Deocamdata este la latitudinea proiectantului specificarea structurilor de testare.