UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Instrumente Structurale
2007-2013

Platformă de e-learning și curriculă e-content pentru învățământul superior tehnic

Testarea Sistemelor

15. GAT pentru circuitele secvenţiale

GENERAREA AUTOMATĂ A TESTELOR PENTRU CIRCUITELE SECVENŢIALE

Modelarea circuitelor secvenţiale prin reţele combinaţionale iterative unidirecţionale

În aceasta secţiune vor fi prezentate modalităţi de folosire a metodelor de generare a testelor pentru circuite combinaţionale, aplicate pentru circuite secvenţiale. Abordarea se va mărgini, pentru început, la circuite secvenţiale sincrone compuse din porţi şi din bistabili sincroni. Extinderea metodelor de generarea testelor de la circuite combinaţionale la circuite secvenţiale sincrone se bazează pe tehnica de modelare, care transformă un circuit secvenţial sincron într-o reţea combinaţională iterativă unidimensională.

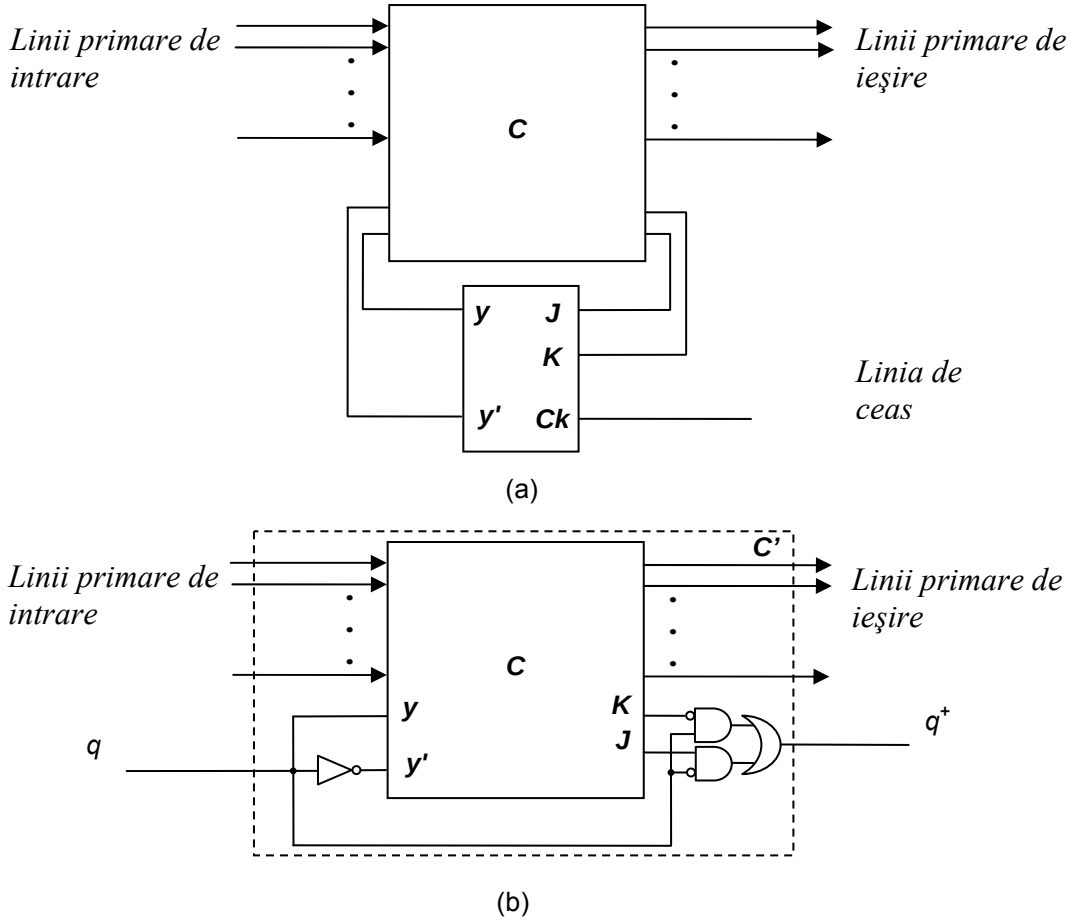


Figura 1. Circuit secvenţial cu bistabil *JK*.
(a) Modelul general.
(b) Modelul pentru un cadru de timp.

O celulă a acestei reţele se numeşte *cadru de timp*. În aceasta transformare un bistabil este modelat ca un element combinaţional care are o intrare suplimentară q faţă de bistabilul pe care-l modelează, intrare care reprezintă starea curentă a bistabilului şi o ieşire adiţională q^+ reprezentând starea viitoare a bistabilului, ieşirea adiţională coincide cu intrarea suplimentară q a

cadrului de timp imediat succesiv.
Un vector de intrare al reţelei iterative combinaţionale reprezintă o secvenţă de intrări aplicate circuitului secvenţial. Se presupune, iniţial, că toţi bistabilii sunt direct conduşi prin aceeaşi linie de ceas, presupusă ca fiind, fără defecte. Prin această ipoteză linia de ceas se poate trata ca o linie implicită în acest model.
În figura 1(b) este arătată structura unui cadru de timp a modelului reţea combinaţională iterativă corespunzător circuitului din figura 1(a) care utilizează un singur bistabil *JK*. Modelul la nivel de poartă pentru bistabilul *JK* implementează ecuaţiile:

$$q^{+} = Jq' + K'q$$
$$y = q$$
$$y' = q'$$

Dacă investigaţia nu focalizează asupra studiului *defectelor interne* ale bistabililor, atunci se poate modela circuitul care realizează funcţia q^{+} ca un modul cu trei intrări : *J, K* şi *q*.
Deoarece toate cadrele de timp au aceeaşi structură, nu este necesar să se construiască modelul complet al reţelei combinaţionale iterative. Algoritmii de generare a testelor pot folosi, astfel, acelaşi model structural pentru fiecare cadru de timp; valorile semnalelor din cadre de timp diferite, totuşi, trebuiesc stocate separat.
Deoarece circuitul *C'*, obţinut prin adăugarea modelului pentru bistabil la circuitul combinaţional original *C*, este combinaţional se pot aplica algoritmii cunoscuţi pentru circuitele combinaţionale. Se consideră, pentru început, un algoritm orientat pe defecte. Un vector de test *t* generat de un astfel de algoritm pentru circuitul *C'* poate specifica valori atât pentru liniile primare de intrare cât şi pentru variabilele *q*; ultimele trebuind sa fie justificate în cadrele de timp precedente. Vectorul de test *t* poate propaga o eroare, similar, nu la o linie primară de ieşire ci la o variabila q^{+}. Atunci eroarea trebuie sa fie propagată în următorul cadru de timp. Astfel procesul căutării poate cuprinde câteva cadre de timp, mergând atât înainte cât şi înapoi, în timp. O dificultate majoră apare din faptul că nu se cunoaşte *a priori* numărul de cadre de timp într-un sens sau altul al axei timpului.
Atunci când sunt mai multe cadre de timp, defectul ţintă este prezent în fiecare cadru de timp. Defectul singular original, cu alte cuvinte, este echivalent unui defect multiplu în modelul reţelei combinaţionale iterative. În consecinţă, o valoare D sau D' se poate propaga chiar peste linia defectă (acest lucru nu poate sa aibă loc într-un circuit combinaţional, cu un singur defect).

Valoare propagată pe linia *w*	Defectul situat pe linia *w*	Valoarea rezultată a liniei *w*
D	*b-l-*0	*D*
D	*b-l-*1	1
D'	*b-l-*0	0
D'	*b-l-*1	*D'*

Figura 2. Rezultatul propagării peste o linie defectă a consecinţelor defectului.

Dacă linia defectă este *b-l-c*, şi valoarea propagată peste această linie este v/v_d, valoarea compozită rezultată este *v/c* (a se vedea figura 2). De remarcat că atunci când *v* = *c*, linia defectă opreşte propagarea consecinţelor defectului generate în cadre de timp anterioare.

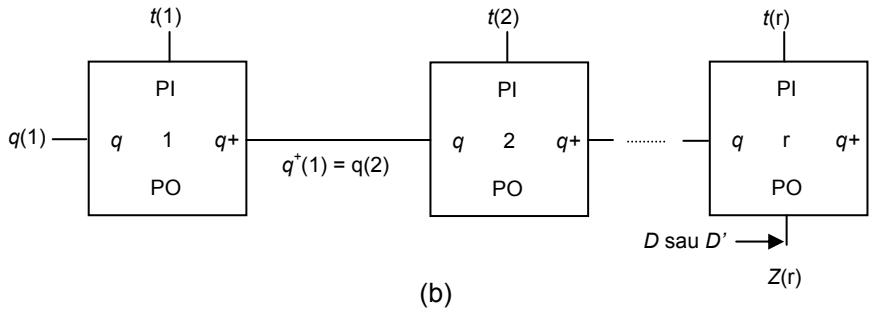
Atunci când se generează o secvenţă de test pentru un defect nu este necesar să se parcurgă aceeaşi stare de două ori. Mai mult, admiterea ca o stare anterior parcursă să fie reparcursă va conduce la o buclă infinită. Pentru a evita astfel de situaţii, orice implementare a unui algoritm de generare a testelor pentru circuitele secvenţiale include un mecanism de supraveghere a stărilor şi de regresie atunci când o stare este repetată.

Generarea algoritmică a testelor pornind dintr-o stare iniţială cunoscută

Forma cea mai generală a unui algoritm de generare a testelor pentru circuitele secvenţiale presupune că starea iniţială a circuitului etalon şi a circuitului testat este necunoscută, deoarece starea bistabililor este arbitrară, atunci când se pune sub tensiune circuitul. Pentru început va fi examinat un algoritm mai puţin general care va presupune că vectorul stării iniţiale $q(1)$ este cunoscut atât în circuitul etalon cât şi în circuitul testat (a se vedea figura 3). În această situaţie procesul de căutare al testului merge doar *înainte în timp*. Deoarece se doreşte generarea cât mai repede cu putinţă a unui test, se va încerca o secvenţă de lungime $r + 1$ numai dacă nu se reuşeşte printr-o secvenţă de lungime r. Astfel, atunci când se consideră o reţea combinaţională iterativă cu $r + 1$ cadre de timp, se vor ignora liniile primare ale primelor r cadre de timp. Deasemenea se vor ignora ieşirile q^+ din ultimul cadru de timp. Odată reţeaua iterativă construită, se poate aplica orice algoritm de generare a testelor pentru circuite combinaţionale exceptând doar unele modificări minore impuse de prezenţa liniei defecte din fiecare cadru de timp.

```
r = 1
repetă
        {
                construieşte modelul cu r cadre de timp;
                ignoră liniile primare de intrare în primele r-1 cadre de timp;
                ignoră liniile q+ din ultimul cadru de timp;
                q(1) = starea iniţială dată;
                dacă(generarea_testului = SUCCES) atunci întoarce SUCCES;
                /* nu există soluţie cu r cadre de timp */
                r = r + 1;
        }
până când  r = f_max
return EŞEC;
```

(a)



(b)

Figura 3. Generarea testelor pornind dintr-o stare iniţială cunoscută.
(a) Procedeul general; (b) Modelul reţea combinaţională iterativă.

O problemă importantă este decizia opririi incrementării numărului de cadre de timp. Cu alte cuvinte, ar trebui să se ştie lungimea maximală a unei posibile secvenţe de test pentru un defect. Astfel pentru un circuit cu n bistabili, pentru fiecare din cele 2^n stări ale circuitului etalon, circuitul defect poate fi, la rândul sau, într-una din cele 2^n stări ale sale. Lungimea unei secvenţe de test ce nu repetă stări, în consecinţă, nu poate fi mai mare decât 4^n. Această limită, totuşi este prea mare pentru o utilitate practică, aşa că se foloseşte limitarea numărului de cadre de timp specificată de utilizator, mult mai mică decât cea teoretică, notată, în algoritmul din figura 3, prin f_{max}.
Cu toate că este corect conceptual, algoritmul schiţat în figura 3 este ineficient, deoarece atunci când se creează reţeaua iterativă combinaţională cu r cadre, toate calculele făcute anterior în primele r-1 cadre sunt ignorate. Pentru a se evita o astfel de risipă, se pot salva secvenţele parţiale care pot fi extinse în cadrele viitoare. Anume, câtă vreme se caută o secvenţă S_r, de lungime r, se salvează orice secvenţă ce propagă erori la variabilele q^+ din cadrul r (şi vectorul de stări rezultat $q^+(r)$) într-un set SOL_r numit astfel pentru că stochează soluţii parţiale.
Dacă nici o secvenţă S_r nu propagă o eroare la o linie primară de ieşire, atunci căutarea pentru o secvenţă S_{r+1} de lungime $r + 1$ începe dintr-una din stările salvate $q^+(r)$.

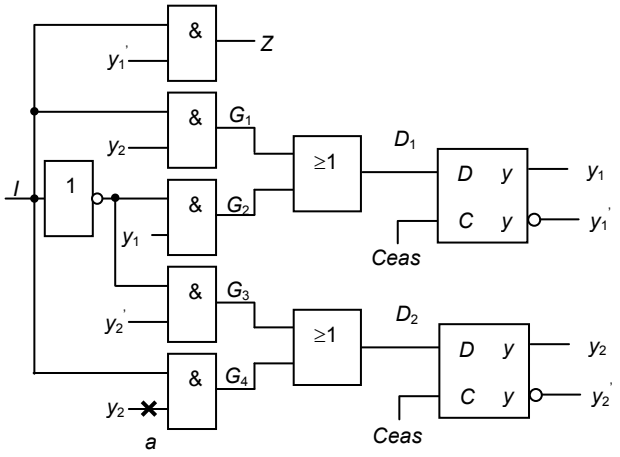


Figura 4.a. Circuitul secvenţial sincron din exemplul 1.

Exemplul 1*:* Se consideră circuitul secvenţial din figura 4(a) şi defectul *a b-l*-1. Se va deduce o secvenţă de test care detectează acest defect, presupunând că starea iniţială este $q_1 = q_2 = 0$. În figura 4(b) este arătat modelul unui cadru de timp.

Cadrul de timp 1*:* Se aplică $q_1 = q_2 = 0$, ceea ce creează un D' în punctul de inserţie al defectului. Vectorul $I(1) = 1$ propagă eroarea la q^+_2. Deoarece consecinţa defectului nu se propagă la linia *Z,* se salvează secvenţa $S_1 = (1)$ şi starea $q^+(1) = (0,D')$ în setul SOL_1.

Cadrul de timp 2: Se aplică $(0, D')$ liniilor q din cadrul 2. Acum *D-frontiera* este $\{G_1, G_3, G_4\}$. Dacă se alege G_1 ori G_4 pentru propagarea erorii, aceasta conduce la $I(2) = 1$ şi starea următoare $q^+(2) = (D', D')$. Încercarea de propagare a erorii prin G_3 rezultă în $I(2) = 0$ şi starea următoare $q^+(2) = (0, D)$.

Cadrul de timp 3: Acum setul SOL_2 conţine două soluţii parţiale:
(1) secvenţa (1, 1) conducând la starea *(D', D')* şi
(2) secvenţa (1, 0) conducând la starea $(0, D)$.

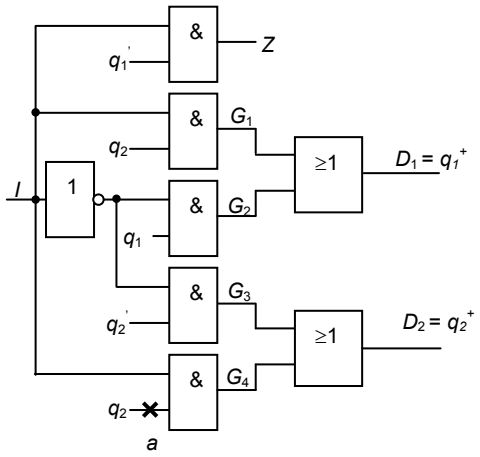


Figura 4.b. Modelul cadrului de timp pentru circuitul secvenţial sincron din exemplul 1.

Încercând prima soluție parțială, se aplică (*D', D*') liniilor *q* din cadrul de timp 3, ceea ce conduce la *D-frontiera* {*Z, G*$_1$*, G*$_2$*, G*$_3$*, G*$_4$}.
Se alege *Z* pentru propagarea erorii şi se obţine *I*(3) = 1 şi *Z* = *D*.
Secvenţa de test rezultată este *I* = (1, 1, 1). □

Se remarcă cu ușurință faptul că dintr-o stare inițială dată, s-ar putea să nu se poată găsi întotdeauna posibilitatea activării defectului, producerii erorii, în prima secvență de timp; în astfel de situații este necesară o secvență de activare a defectului.

Reutilizarea unor probleme anterior rezolvate.
O dificultate suplimentara în generarea testelor (GT) pentru circuitele secvenţiale este existenţa unor stări "imposibile", adică a unor stări niciodată folosite în operarea normală, liberă de defecte, a unor astfel de circuite. Spre exemplu, un numărător cu cinci stări implementat prin trei circuite bistabile poate avea trei stări în mod normal niciodată folosite. Încercarea de justificare a unor astfel de stări invalide pe durata GT este limitată şi abandonată dar consumă un timp considerabil de calcul. Pentru a ajuta la evitarea unor astfel de probleme, se poate construi un tabel de stări invalide şi nejustificabile. Inițial, tabelul va fi completat de utilizator. Apoi sistemul va adăuga stările a căror justificare a fost imposibilă sau care nu s-au putut justifica în limita de timp specificată de utilizator. Înainte să încerce o stare **q**, sistemul de GT va compara această stare cu stările din tabel; în cazul în care comparația conduce la concluzia că starea **q** este acoperită de o stare invalidă se inițiază regresia, *backtraking*-ul. Spre exemplu, dacă starea 0xx1 nu este justificabilă (în limitele permise), atunci este inutil să se încerce justificarea stării 01x1, care este, evident, o restricție mai stringentă în spațiul stărilor.

Acest concept al reținerii unor probleme anterioare de justificare a stărilor poate fi extins şi la probleme rezolvate favorabil. Spre exemplu, se presupune ca se dorește justificarea stării **q** = 0x1x, iar starea 011x a fost anterior justificată. Evident, se poate aplica din nou secvența folosită să se justifice o stare acoperită de **q**. Se poate construi un tabel de probleme de justificare a stărilor, rezolvate, împreuna cu soluțiile corespunzătoare. Ori de câte ori se cere justificarea unei stări **q**, se caută în tabel să se verifice dacă stare **q** (sau o stare acoperită de aceasta) a fost justificată anterior. În caz afirmativ secvența de justificare este reutilizată.

Funcţiile cost

Funcţiile cost observabilitate şi controlabilitate introduse iniţial la GT pentru circuitele combinaţionale sunt fost extinse şi asupra circuitelor secvenţiale. Calculul costurilor controlabilitate porneşte prin atribuirea $C0(l)=C1(l)=\infty$ pentru fiecare linie l (în practică, ∞ este reprezentat de un număr întreg cu o valoare foarte mare). Apoi valorile $C0(i)$, $C1(i)$ ale fiecărei LPI i sunt schimbate la valoarea f_i-1, unde f_i este factorul de ramificare al LPI i (folosind funcţii cost bazate pe ramificaţii). Apoi *schimbările costurilor* sunt propagate mai departe în aceeaşi maniera în care evenimentele logice sunt propagate atunci când se simulează circuitele logice. Calculul noului cost al ieşirii unei porţi ca urmare a schimbării costurilor intrărilor porţii respective se face cu formulele obişnuite. Se remarcă, pentru o poartă AND p, $C1(p)$ va lua valoarea infinit până când fiecare intrare a porţii p va avea o valoare finita pentru $C1$, controlabilitatea în 1 a respectivei intrări. Bistabilii sunt procesaţi în baza ecuaţiei lor de funcţionare. Costurile de controlabilitate ale variabilelor de stare şi a liniilor afectate de acestea pot să se schimbe de câteva ori pe durata acestui proces, până când toate costurile ajung să-şi atingă valorile lor stabile. Convergenţa procesului de calcul este garantată deoarece atunci când un cost îşi schimba valoarea, întotdeauna valoarea acestuia scade. În mod uzual costurile îşi ating valoarea stabilă în numai câteva iteraţii.

După determinarea costurilor de controlabilitate, sunt calculate costurile de observabilitate printr-un proces iterativ similar. Se porneşte cu iniţializarea $O(l)=\infty$ pentru fiecare linie l din circuit şi cu iniţializarea $O(LPE)=0$. În continuare, schimbările costurilor de observabilitate sunt propagate înapoi prin circuit, aplicând formulele obişnuite.

Alegerea defectelor ţintă

Structura unui sistem de GT pentru circuite secvenţiale folosind algoritmi de GT orientaţi pe defecte este similară unuia dedicat circuitelor combinaţionale, exceptând faptul ca în cazul de faţă este generată o secvenţă de test în locul unui singur vector de test pentru detecţia unui defect. Pentru orice defect ţintă GT porneşte de la starea existentă la sfârşitul secvenţei anterior generate.

Fiecare secvenţă generată este simulată pentru a se determina toate defectele pe care aceasta le detectează. Un co-produs important al acestei simulări a defectelor este acela că se determină deasemenea defectele nedetectate ale căror efecte s-au propagat la variabilele de stare. Această informaţie este folosita prin selecţia unui astfel de defect ca ţinta următoare; o astfel de alegere "oportuna" scuteşte efortul activării defectului şi efortul propagării unei erori către o variabilă de stare.

Logica asociată semnalului de ceas şi circuitele cu ceasuri multiple

Până acum s-a presupus ca toate bistabilele sunt direct acţionate de aceeaşi linie de ceas liberă de defecte. Totuşi, în practica, circuitele secvenţiale pot avea linii de ceas multiple iar ceasurile se pot propaga printr-o logică (sub-circuit) în calea lor către bistabili.

Consideraţii privind complexitatea generării algoritmice a testelor pentru circuitele secvenţiale

Generarea testelor pentru circuitele secvenţiale este mult mai complexă comparativ cu problema similară pentru circuitele combinaţionale, în principiu, pentru că atât justificarea liniilor cât şi propagarea erorii implică cadre de timp multiple. În cazul cel mai defavorabil, numărul necesar de cadre de timp este o funcţie exponenţială de numărul de bistabili.

O proprietate structurală importantă a circuitelor secvenţiale este prezenţa unor *cicluri*, unde un ciclu este o buclă cuprinzând un bistabil. Generarea testelor pentru un circuit secvenţial fără

cicluri nu este mult mai dificilă decât cea pentru un circuit combinaţional comparabil. Complexitatea unei generări de teste pentru circuite secvenţiale cu cicluri este direct corelată cu structura sa ciclica, aceasta putând fi caracterizată prin:
- numărul de cicluri;
- numărul de bistabili dintr-un ciclu;
- numărul de cicluri în care este prins un bistabil (reflectă gradul de interacţiune dintre cicluri).

În general nu se poate aştepta ca generarea testelor pentru circuite secvenţiale mari cu structură ciclică complexă, să fie fezabilă. Soluţiile practice se bazează pe o proiectare specifică care include tehnici de testabilitate. Astfel de tehnici transformă un circuit secvenţial astfel încât testarea unui circuit secvenţial se apropie mult de testarea unui circuit combinaţional pe durata testării.

Circuitele secvenţiale asincrone

GT pentru circuitele secvenţiale asincrone este considerabil mult mai dificila decât pentru circuitele secvenţiale sincrone. Întâi, circuitele asincrone conţin adesea curse şi sunt susceptibile la operării improprii cauzate de hazarduri. Apoi, pentru obţinerea unui model reţea iterativa combinaţională a circuitului, este necesar sa se identifice toate liniile de reacţie din respectivul circuit; acest lucru este o sarcina complexa de calculat. În fine, operarea corectă a circuitelor asincrone depinde adesea de întârzierile intenţionat plasate prin circuit, dar nici unul dintre algoritmii de GT anterior discutat nu lua în calcul şi întârzierile.

Se remarcă faptul că un circuit asincron poate sa treacă printr-o secvenţa de stări ca răspuns la o schimbare a intrărilor. Se presupune ca valorile LPE sunt măsurate abia după atingerea stării stabile şi se presupune deasemenea ca LPI nu îşi schimbă valorile înainte ca respectivul circuit să se fi stabilizat.

2. Generarea testelor bazata pe simulare pentru circuitele secvenţiale

Principiul GT bazate pe simulare este explorarea spaţiului vectorilor de intrare:
- Se generează şi se simulează (cu defecte) vectorii de test;
- În baza rezultatelor obţinute prin simulare, se evaluează vectorii de test;
- Se aleg cei mai "buni" vectori de test în raport cu anumite funcţii de cost;

Iniţial se porneşte cu un vector arbitrar (sau specificat de utilizator). Vectorii de test sunt de regula aleşi dintre "*vecinii*" vectorului de intrare *t*, curent, adică vectori diferiţi de *t* printr-un singur bit. Procesul de alegere este aleator, aşa ca aceasta metoda este oarecum similara generării aleatoare a testelor. Funcţia cost depinde de obiectivele fazei curente a procesului de GT:
1. *Iniţializarea:* aici obiectivul este aducerea tuturor bistabililor la valori binare. Funcţia cost este numărul de bistabili aflaţi în stare necunoscută. (Această fază nu necesită simularea defectelor.)
2. *Generarea testelor pentru grupuri de defecte:* în aceasta faza obiectivul este detectarea a cât mai multe defecte cu putinţă, deci toate defectele nedetectate sunt simulate. Pentru fiecare defect activat *i* se calculează costul acestuia c_i ca fiind cea mai scurta distanta dintre porţile din *D-frontiera* sa şi LPE. Distanţa este numărul ponderat de elemente de pe o cale, bistabilii având ponderi mai mari decât porţile. Costul c_i reprezintă o măsură a "cât de aproape este detectarea defectului *i*". Funcţia cost este suma costurilor c_i cu *i* din mulţimea indicilor unui subset ce conţine defectele cu cele mai mici costuri.
3. *Generarea testului pentru defecte individuale:* se urmăreşte în aceasta faza unul dintre defectele ramase nedetectate iar obiectivul este generarea unei secvenţe de test care sa-l detecteze. Funcţia cost este o *măsură dinamică a testabilităţii*, aleasă să măsoare efortul suplimentar necesar sa se detecteze defectul atunci când se porneşte din starea curenta.

În toate cele trei faze, un "bun" vector de test va reduce costul. Pe durata fazei de inițializare, reducerea costului semnifica inițializarea mai multor bistabili. Atunci când costul devine zero, toți bistabilii au valori binare. În celelalte doua faze, reducerea costului înseamnă aducerea unui grup de defecte sau a unui defect individual, aproape de detecție. În principiu, ar trebui sa fie evaluaţi toți vectorii de test şi sa se aleagă acela ce conduce la prețul cel mai mic. (Într-un circuit cu *n* LPI, se pot obține *n* vectori de test prin schimbarea a câte un singur bit din vectorul curent.) În practică se poate aplica o strategie "*greedy*", care acceptă primul vector ce reduce costul. Rejectarea unui vector de test care nu reduce costul înseamnă restaurarea stării problemei (valori, liste de defecte etc.) corespunzătoare celei create de ultimul vector acceptat; acest proces este similar unei regresări, "*backtracking*". Uneori se poate întâmpla ca nici un vector de test sa nu poată să reducă în continuare costul, ceea ce arata ca procesul de căutare a atins un minimum local. În acest caz este necesar sa se schimbe strategia, astfel: (1) se accepta un vector ce creşte costul sau,
(2) se generează un nou vector care diferă de cel curent prin mai mult decât un bit sau,
(3) se trece la faza următoare.

Un avantaj important al GT bazate pe simulare este acela ca se poate folosi acelaşi model al întârzierii ca şi simulatorul de defecte. Se pot aborda astfel circuite asincrone. Deoarece componenta de baza este un simulator de defecte convențional, GT bazata pe simulare este mai ușor de implementat decât metodele ce utilizează modelul iterativ.

3 Generarea testelor folosind modele RTL

Generarea testelor descrise în paragrafele anterioare se aplica acelor modele de circuite ce sunt alcătuite din porți şi bistabili. În general un proiectant sau un inginer de teste uzual percepe circuitul ca o interconectare de componente de nivel înalt, cum ar fi numărătoare, registre şi multiplexoare. Cunoaşterea modului cum operează aceste componente permite soluții simple şi compacte pentru problemele de justificare a unor linii şi pentru problemele de propagare a erorilor. Spre exemplu, justificarea unui 1 în bitul al doilea al unui numărător aflat curent în starea cu toți biții în 0, este făcută prin executarea a doua operații de incrementare. În schimb un program ce lucrează cu un model structural de nivel coborât (în care numărătorul este interconectarea unor porți şi bistabili) nu poate folosi acest tip de cunoaştere funcțională.

Principala motivație pentru modelele RTL este creşterea vitezei de GT pentru circuitele secvențiale simultan cu creşterea mărimii circuitelor ce pot fi procesate. O alta motivație este folosirea curentă a unor componente comerciale pentru care modelele la nivel de poartă nu sunt cunoscute, dar ale căror descrieri RTL sunt cunoscute sau pot fi ușor deduse din specificația lor funcțională.

4 Concluzii

GT eficienta pentru circuite mari este o problema practica deosebi de importanta. Câteva dintre noile direcții de cercetare din acest domeniu implica suport hardware şi tehnici de Inteligenta Artificiala (IA).

Suportul hardware pentru GT poate fi realizat prin arhitecturi speciale de calculatoare folosind tehnici de conductă (*pipeline*) sau prin arhitecturi multiprocesor de uz general. Arhitecturile multiprocesor pot folosi paralelismul disponibil propriu lor în anumite maniere specifice GT:

- *procesarea concurenta a defectelor*, unde fiecare procesor executa aceeași sarcina de GT dar pentru un subset separat al setului general al defectelor;
- *procesarea concurenta a deciziilor*, unde fiecare procesor explorează simultan ramuri diferite pornind din același nod al arborelui de decizie;
- *procesarea concurenta ghidata*, unde diferite procesoare țintesc același defect dar folosind funcții cost diferite pentru ghidare.

Tehnicile de IA sunt aplicate în GT ca o încercare de a emula trăsături, caracteristici, ale raționamentelor inginerilor experți în testare atunci când aceştia procedează la examinarea unei probleme de GT. Spre deosebire de un algoritm de generare automata a testelor, care este insensibil la funcționalitatea circuitului şi urmărește numai o porțiune îngustă a structurii sale la un moment dat, un inginer expert în testare se bazează în special pe cunoașterea la nivel ridicat a comportamentului intenționat a circuitului şi a componentelor sale şi pornește de la un punct de vedere global asupra întregii structuri ierarhice din problema examinata. O altă diferență este abilitatea unui astfel de inginer în recunoașterea unei familii de probleme similare şi capacitatea acestuia să tragă un avantaj din similaritatea lor prin găsirea unor procedee de soluționare comune; spre exemplu, aceeași secvență de control pentru încărcarea un registru poate fi repetată cu diferite date la momente arbitrare. Un algoritm de GT ar regenera în mod "mecanic" mereu aceeași secvență de control la fiecare moment când trebuiesc încărcate respectivele date în registru.

Faţă de un algoritm de GT, care explorează spațiul operațiilor posibile ale unui circuit, un inginer expert în testare cercetează în spațiul mult mai restrâns al operațiilor intenționate, adică: operațiile pentru care a fost proiectat circuitul sa funcționeze. În acest sens exista sisteme (M. Shirley, P. Wu, R. Davis, şi G. Robinson, 1987) ce folosesc cunoașterea comportamentului intenționat al unui circuit pentru reducerea domeniului de căutare al soluțiilor.

În loc sa fie furnizată de utilizator, cunoașterea este realizata prin analiza trasării obținute pe parcursul *simulării simbolice* a circuitului. Un simulator simbolic, în plus de propagarea valorilor constante (lucru pe care-l face şi un simulator obişnuit), propaga şi variabile iar rezultatele sale sunt expresii ce descriu comportamentul circuitului pentru toate datele posibile reprezentate de variabilele respective. Spre exemplu, daca X şi Y sunt variabile atribuite intrărilor A şi B ale unui sumator, atunci ieşirea sumatorului va fi expresia $X+Y$, şi daca $X=0$, atunci ieşirea va fi Y. Pe parcursul generării testelor, obiectivele căutate sunt comparate şi "potrivite" cu trasările simulării simbolice. Astfel, daca pentru sumator obiectivul este să se propage o eroare aflata la intrarea B, atunci soluția $A = 0$ va fi găsită printre trasările simulării simbolice care vor stoca deasemenea şi evenimentele care au condus la $A = 0$; aceste evenimente sunt recuperate şi reutilizate ca părți ale testului generat.

Tehnicile de GT au şi alte aplicații în afara scopului inițial. Poate apare situația în care exista spre exemplu, doua circuite combinaționale ce se presupune ca implementează aceeași funcție. Sa presupunem ca, spre exemplu, circuitul C_1 este obținut cu o proiectare manuala şi circuitul C_2 este obținut prin sinteza automata, amândouă pornind de la aceleași specificații. O situație similara poate fi, în același context, în care C_1 este un circuit existent iar C_2 este o reproiectare a circuitului C_1 folosind o tehnologie diferita. Se pune problema sa se compare cele doua circuite C_1 şi C_2.

Bineînțeles se pot simula circuitele C_1 şi C_2 cu aceiași stimuli şi se pot compara rezultatele, dar de regula procesele de simulare (din cauza volumului adesea foarte mare de vectori posibili de intrare) nu oferă o verificare completa. *Verificarea logică* poate compara automat cele două

circuite şi dovedi fie că sunt implementarea aceleiași funcții, fie să genereze vectori ce cauzează rezultate diferite în cele două circuite. Abordarea de bază descrisa prima data de J.P. Roth în 1977, constă din combinarea celor doua circuite C_1 şi C_2 într-un singur circuit compozit M şi apoi aplicarea unui algoritm de justificare a unei valori 1 pe rând la fiecare LPE a circuitului M. Dacă justificarea se dovedește posibilă atunci vectorul generat aduce ieșirile corespunzătoare circuitelor C_1 şi C_2 la valori diferite şi în consecință cele doua circuite nu realizează aceeași funcție; altfel nu exista un astfel de vector şi cele doua LPE au aceeași funcție.