UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Instrumente Structurale 2007-2013

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

Testarea sistemelor

1. Modelarea funcţională la nivel logic şi la nivel RTL, modelele structurale

Manufactura unui produs numeric (digital) constă din fabricarea şi testarea acestuia. Proiectarea produsului şi dezvoltarea unor teste (secvenţe de vectori de test) preced manufactura. Proiectarea este - într-un fel - o sinteză a detaliilor manufacturii în timp ce dezvoltarea unui test specifică datele testului dar şi detaliile procedeelor de testare. Proiectarea leagă specificaţiile abstracte de specificaţiile fizice ale produsului printr-un proces de sinteză sau de asamblare ale unor componente existente şi, de asemenea, generează datele necesare conducerii echipamentelor care produc fizic respectivului produs. Verificarea, constând din analiză şi simulare, cercetează corectitudinea proiectării. Corectitudinea funcţionării produsului fabricat este determinată prin testare. Astfel, pentru realizarea unui produs funţionând corect, sunt necesare atât proiectarea cât şi testarea acestuia.

Testare şi Diagnoză

Definiţii

Testarea unui sistem este constituită dintr-un şir de experimente prin care se exercită respectivul sistem în aşa fel încât răspunsul sistemului, ca urmare a şirului de experimente, să fie suficient şi necesar pentru a stabili dacă acesta funcţionează corect sau nu.

In cazul în care se stabileşte că sistemul funcţionează incorect se poate formula o a doua cerinţă a testării: diagnoza; aceasta însemnând localizarea sau evidenţierea cauzei care a condus la respectiva malfuncţionare a sistemului. Diagnoza presupune, evident, cunoaşterea structurii interne a sistemului testat.

Testarea la diferite nivele de abstractizare

Deoarece consideraţiile anterioare pot fi aserţiuni valabile pentru testarea sistemelor în general, în cele ce urmează se va restrânge aria de consideraţii numai asupra sistemelor şi subsistemelor numerice (digitale).
Denumirile de sistem şi subsistem, exceptând situaţiile în care este evidentă o ierarhizare, vor fi utilizate alternativ - întrucât un sistem poate fi întotdeauna considerat un subsistem al altui sistem.
O caracteristică importantă a sistemelor digitale o constituie complexitatea; de complexitatea unui sistem este strâns corelat gradul de abstractizare al considerării acestuia.
Nivelul de abstractizare al descrierii unui sistem poate fi succint caracterizat prin tipul de informaţie procesat de respectivul sistem (a se vedea figura 1.1)

Control	Date	Nivel de abstractizare
1. Valori Logice (sau secvenţe de)	Valori Logice (sau secvenţe de)	Nivel Logic
2. Valori Logice	Cuvinte	Nivel Registru
3. Instrucţiuni	Cuvinte	Nivel Set de Instrucţiuni
4. Programe	Structuri de Date	Nivel Procesor
5. Mesaje	Mesaje	Nivel Sistem

Figura 1.1 Nivele de abstractizare ale informaţiei procesate de un sistem.

Nivelul Logic

Informaţia procesată la acest nivel poate fi reprezentată prin valori logice discrete: de regulă valori logice binare (0 şi 1) Nivele de abstractizare (sau modele) mai rafinate pot reclama mai multe valori logice.

O distincţie mai profundă în acest sens poate fi făcută ţinând seama de natura combinaţională (nu reflectă o istorie anterioară a sistemului) sau secvenţială (răspunsul sistemului la un moment dat depinde atât de stimulii anterior aplicaţi cât şi de stimulii actual aplicaţi) a comportamentului sistemului.

De regulă sistemele considerate la acest nivel sunt circuitele; orice sistem poate fi considerat la nivel de circuit dar efortul de analiză poate fi mult prea mare sau chiar imposibil.

Nivelele de abstractizare superioare nivelului logic sunt folosite atunci când considerarea operaţiilor efectuate de un sistem la acest nivel conduce la modele greoaie, stufoase şi foarte dificil de folosit.

Nivelul Registru

În mod uzual un sistem este conceput prin interacţiunea informaţiilor procesate (datele) cu informaţia de stare (control) a sistemului. Menţinând definiţia funcţiei de control la nivel de valori logice, în cadrul acestui nivel se consideră că informaţia procesată (datele) este grupată în vectori (cuvinte) de valori logice. Deoarece cuvintele sunt stocate în decursul procesării în registre, acest nivel a primit numele corespunzător.

Nivelul Setului de Instrucţiuni

Nivelul acesta de abstractizare consideră informaţia de control grupată în cuvinte numite instrucţiuni.

Nivelul Procesor

Acest nivel, imediat superior celui anterior, consideră un sistem numeric ca procesând secvenţe de instrucţiuni, sau programe, care operează asupra unor blocuri de date numite structuri de date.

Nivelul Sistem

Un mod diferit de a vedea un sistem numeric (nu neaparat la un nivel mai ridicat de abstractizare) este considerarea acestuia ca fiind constituit din mai multe subsisteme independente, sau unităţi, ce inter-comunică prin blocuri de cuvinte numite mesaje.

In general, stimulii şi răspunsul la aceşti stimuli ai unui sistem oarecare, sunt elementele care definesc un experiment de testare şi corespund tipului de informaţie procesat de sistemul aflat în test (abrevierea curentă este SAT). Din acest punct de vedere testarea este un termen generic care acoperă un spectru larg de medii şi activităţi:

- unul sau mai multe subsisteme care testează un altul prin transmiterea şi recepţia unor mesaje;

- un procesor care se autotestează prin execuţia unui program de diagnostic;

- un echipament automat de testare (EAT) care verifică un circuit prin aplicarea unor vectori binari de test, operaţie conjugată cu examinarea vectorilor binari emişi de respectivul circuit ca urmare a stimulilor aplicaţi.

O altă categorie de teste sunt testele parametrice. Această testare tratează problema examinării caracteristicilor electrice ale circuitelor (cum ar fi tensiunile de polarizare, curenţii de pierderi, tensiunile de prag, etc.) . Aceste tipuri de testări nu fac obiectul acestui curs.

Erori şi Defecte

O apariţie (instanţiere) a unei malfuncţionări a unui sistem testat (sau **UAT,** prescurtare pentru ***Unitate Aflată în Test*)** se numeşte *eroare* (observată). De remarcat faptul că în general conceptul de eroare are interpretări diferite la nivele de abstractizare distincte.

Spre exemplu, o eroare observată la nivelul unui program de diagnostic poate apare ca un

rezultat incorect al unei operaţii aritmetice (să admitem), în timp ce pentru un EAT o eroare înseamnă de obicei o valoare binară incorectă.

Cauzele unei erori observate pot consta din: erori de proiectare, erori de fabricaţie, defecte de fabricaţie şi defecte fizice.

Exemple tipice de erori de proiectare:

- specificaţii incomplete şi/sau contradictorii;

- corespondenţe si transformări eronate între nivele succesive de proiectare;

- nerespectarea regulilor de proiectare.

Printre erorile care au loc pe durata fabricaţiei sunt cuprinse:

- componente necorespunzătoare;

- conexiuni incorecte;

- scurtcircuite datorate unor conexiuni sudate sau metalizate inadecvat.

Defectele de fabricaţie nu sunt direct atribuibile erorilor umane; mai degrabă aceste defecte rezultă dintr-un proces de manufacturare imperfect. Spre exemplu scurtcircuitele (adeseori prescurtate "scurturi") şi întreruperile sunt defecte comune în fabricaţia circuitelor MOS pe Scară Largă de Integrare (se va folosi, în continuare, totuşi prescurtarea MOS LSI, devenită clasică). Alte defecte de fabricaţie includ profile de dopaj necorespunzătoare, erori de aliniere ale măştilor şi încapsulări defectuoase. O precisă localizare a defectelor de fabricaţie este importantă pentru creşterea productivităţii manufacturării respective.

Defectele fizice apar pe durata timpului de viaţă al sistemului şi sunt datorate degradării componentelor şi/sau factorilor de mediu. Spre exemplu, conexiunile de aluminiu din interiorul circuitelor integrate (CI) se subţiază cu timpul (datorita migraţiei electronilor şi coroziunii) şi pot să se întrerupă. Factorii de mediu cum ar fi temperatura, umiditatea şi vibraţiile, accelerează îmbătrânirea componentelor. Radiaţii cosmice şi particule alfa pot induce defecte în chip-urile care conţin memorii de mare densitate cu acces aleator (RAM-uri).

Erorile de proiectare, defectele de fabricaţie şi defectele fizice sunt la un loc referite ca *defecte fizice.* In raport cu stabilitatea lor în timp, defectele pot fi clasificate ca fiind:

- permanente, adică fiind prezente întotdeauna după apariţia lor;

- intermitente, adică au existenţa mărginită numai pe durata unor intervale de timp;

- tranzitorii, adică apariţii singulare cauzate, de regulă, de schimbarea temporară a unui factor de mediu.

In general, defectele fizice nu permit o tratare matematică, o modelare, a testării si diagnozei. O

soluţionare pragmatică a acestei situaţii se realizează cu ajutorul modelării defectelor fizice prin *defecte logice.* Aceasta abordare oferă, în fapt, o reprezentare convenabilă a efectelor defectelor fizice asupra operării normale a unui sistem. Intr-un subsistem / sistem numeric, un defect este *detectat* prin observarea erorii /erorilor cauzate de acesta.

Ipotezele de bază privitor la natura defectelor logice sunt referite generic ca fiind *modelul defectului.* Modelul de defect cel mai larg utilizat este acela al unei linii singulare (fir, conexiune, legătură) care este permanent "blocată" (*stuck-at*, în literatura anglo-americană de profil) la o valoare logică (1 sau 0) . Notaţia curentă pentru o linie arbitrară *w* este : *w blocată-la-0,* sau *w blocată-la-1* (abrevierile curente sunt *w b-1-0,* respectiv *w b-l-1*).

Modelare şi Simulare

Deoarece erorile de proiectare preced fabricaţia unui sistem, se poate realiza *testarea verificării unui proiect* printr-un experiment de test care foloseşte un model al sistemului proiectat. In acest context, "model" înseamnă o reprezentare numerică pe calculator în termeni de structuri de date şi/sau programe.

Modelul poate fi exercitat prin simularea acestuia cu o reprezentare a semnalelor de intrare. Procesul se numeşte *simulare logică* (de asemenea se mai spune *simularea verificării proiectării* sau *simularea valorii adevărate)*. Simularea logică determină evoluţia în timp a semnalelor din model ca răspuns la secvenţele de intrare aplicate.

Evaluarea Testului

O problemă importantă în testare este *evaluarea testului,* prin aceasta înţelegându-se determinarea eficienţei, sau calitatea, unui test. Evaluarea testului se face uzual în contextul unui model al defectului, iar calitatea unui test este măsurată prin raportul dintre numărul de defecte detectate şi numărul total de defecte din universul asumat de defecte, pentru sistemul respectiv; acest raport este referit sub denumirea de *acoperirea defectelor.* Evaluarea testului (sau calificarea testului) este făcută printr-un experiment de testare simulat, numit *simularea* defectului, care calculează răspunsul circuitului (sistemului) în prezenţa defectelor pentru care testul este evaluat. Un defect este detectat dacă există o diferenţă, o deosebire, între răspunsul circuitului (sistemului) liber de defecte - adică, circuitul (sistemul) funcţionând normal - şi răspunsul circuitului (sistemului) afectat de defectul respectiv.

In acest sens se spune că un circuit (sistem) este *redundant* dacă există cel puţin un defect care nu poate fi pus în evidenţă atunci când i se aplica respectivului circuit (sistem) la intrare orice secvenţă posibilă, admisă, de stimuli în conformitate cu specificaţiile respective de funcţionare.

Tipuri de testare

Metodele de testare pot fi clasificate în raport cu mai multe criterii.

Testarea prin *programe de diagnoză,* se realizează deconectat ("*off-line*"), la viteza normală de funcţionare, şi la nivelul sistemului. Stimulii îşi au originea în interiorul sistemului însuşi, când lucrează în regim de auto-testare. În sistemele ale căror logică de control este microprogramată, programele de diagnostic pot fi deasemenea microprograme (micro-diagnosticări).

Anumite părţi ale sistemului, numite global *inima (sâmburele) sistemului,* trebuie să fie libere de defecte pentru a putea permite rularea programelor. Stimulii sunt generaţi prin software sau prin *firmware* şi pot fi aplicaţi adaptiv. Programele de diagnostic sunt de regulă rulate în testarea de întreţinere (preventivă) sau de rutină.

Emularea în circuit, este o metodă de testare care elimină necesitatea unei părţi perfect funcţionale a sistemului testat pentru rularea programelor de diagnostic.

Aceasta metodă este folosită în testarea plăcilor şi sistemelor cu microprocesoare, şi se bazează pe îndepărtarea microprocesorului de pe placă, pe durata testării, şi pe accesarea conexiunilor microprocesorului cu restul sistemului, circuitului, testat.

Testorul poate emula funcţia microprocesorului îndepărtat (de regulă printr-un microprocesor de acelaşi tip). Această configuraţie permite rularea programelor de diagnostic folosind memoria şi microprocesorul testorului.

In *testarea "on-line",* stimulii şi răspunsurile sistemului sunt cunoscute dinainte, deoarece stimulii sunt furnizaţi prin secvenţe recepţionate pe durata modului normal de operare.

Obiectul de interes în testarea "on-line" constă nu atât din răspunsul însuşi, cât din unele proprietăţi ale răspunsului, proprietăţi care ar trebui să rămână invariante pe durata unei funcţionări normale. Spre exemplu, o singură ieşire a unui multiplexor liber de defecte trebuie sa aibă valoarea logică 1.

Scopul unui bit de paritate este crearea unui proprietăţi invariante uşor de verificat. Bitul de paritate este *redundant,* în sensul că acesta nu poartă nici o informaţie strict utilă funcţionării normale a sistemului.

Acest tip de *informaţie redundantă* este caracteristică pentru sisteme care folosesc *coduri detectoare şi coduri corectoare de erori.*

Un alt tip de proiectare fiabilă bazată pe redundanţa este *redundanţa modulară,* care se bazează pe replicarea de un număr de ori a sistemului sau a unei părţi din acesta.

Modulele replicate (acestea trebuie să aibă aceeaşi funcţionare, posibil cu diferite implementări) lucrează cu acelaşi set de intrări, iar proprietatea invariantă este aceea că toate modulele replicate trebuie să producă acelaşi răspuns.

Sistemele cu autoverificare au sub-circuite speciale, numite verificatoare *("checkers"* în engleză), dedicate testării proprietăţilor invariante.

Testarea cu probă ghidată este o tehnica folosită în testarea la nivel de placă. Dacă sunt detectate erori pe durata testării, iniţial la pinii conectorilor (faza a testării cunoscută adesea sub numele de test *merge / nu merge*), testorul decide care linie internă a circuitului să fie monitorizată şi instruieşte operatorul să plaseze o probă (sondă) pe linia selectată. Apoi se reaplică testul.

Principiul este să se urmărească regresiv, propagarea erorii (erorilor) de-a lungul unei căi prin circuit. După fiecare aplicare a testului, testerul verifică rezultatul obţinut pe linia monitorizată şi determină când a fost atins locul implantării defectului şi/sau când se continuă urmărirea regresivă.

În loc să monitorizeze o singură linie la un moment dat, unele testoare pot monitoriza un grup de linii, de regulă pinii unui CI.

Testarea cu probă ghidată este o procedură secvenţială de diagnostic, în care un subset al liniilor interne accesibile este monitorizat la fiecare pas. Anumite testoare folosesc un dispozitiv special de fixare numit *pat-de-cuie ("bed-of-nails"* termenul original) care permite monitorizarea tuturor liniilor interne accesibile ale circuitului, într-un singur pas.

Testarea in circuit, are drept scop verificarea componentelor deja plantate pe placă. Un tester exterior foloseşte un conector cleşte ("clip") de circuit integrat (CI) pentru ca să se poată aplica vectorii de test direct la intrările unui CI şi pentru a-i observa răspunsul.
Testerul trebuie să fie capabil să izoleze electronic CI testat de restul plăcii pe care se află plantat, altfel poate supraîncărca ieşirile altor CI conexe.

Testarea algoritmică se referă la generarea vectorilor de intrare pe durata testării. Numărătoare şi registre de deplasare cu reacţie sunt exemple clasice de subsisteme folosite în generarea stimulilor de intrare.
Generarea algoritmică a vectorilor de test este capacitatea anumitor testere de a produce combinaţii ale unor vectori fixaţi.
Combinaţia dorită este determinată printr-un program de control scris într-un limbaj orientat pe aplicaţie.
Răspunsul aşteptat poate fi generat pe durata testării fie de la un exemplar copie al UAT (unităţii aflate in test) cunoscut ca fiind bun funcţional - aşa-zisa unitate etalon - sau folosind o simulare în timp real a UAT.
Acest tip de testare este numit uneori *testare prin comparaţie,* ceea ce constituie cumva **o** denumire improprie, deoarece compararea răspunsului aşteptat cu un răspuns corect este inerentă oricărei metode de testare.

Metodele bazate pe verificarea anumitor funcţii *t(R)* deduse din *R* - răspunsul UAT- în loc să se folosească chiar *R,* se spune că realizează *testarea compactă,* iar *t(R)* se spune că este reprezentarea *comprimată,* sau *semnătura* răspunsului *R.*

Se pot contoriza, spre exemplu, numărul de valori 1 (sau numărul de tranziţii din 1 în 0, sau din 0 în 1) obţinut la ieşirea circuitului şi apoi se face o comparaţie a acestuia cu numărul aşteptat (calculat pentru circuitul liber de defecte).

Un astfel de procedeu de testare simplifică foarte mult procesul de testare, deoarece în loc să se compare - bit cu bit - întreaga ieşire *R* a circuitului cu ieşirea aşteptată se compară doar reprezentările comprimate sau semnăturile acestor ieşiri.

Astfel, sunt micşorate semnificativ cerinţele de memorie ale testerului, deoarece nu mai este necesar să se stocheze întregul răspuns aşteptat (martorul) iar timpul este, evident, mult scurtat.

Tehnicile de comprimare sunt folosite în mod deosebit în circuitele cu autotestare, unde implementarea funcţiei *t(R)* se face printr-un subsistem special, adăugat suplimentar circuitului. Circuitele cu autotestare au de asemenea un subsistem adiţional dedicat generării stimulilor.

Diagnostic şi Reparaţie

Dacă o UAT care se dovedeşte că funcţionează incorect trebuie să fie reparată, este necesară diagnosticarea cauzei erorii observate. Într-un sens mai larg, termenii diagnostic şi reparaţie se aplică atât defectelor fizice cât şi erorilor de proiectare (pentru cazul din urmă reparaţie însemnând reproiectare) .

Totuşi, câtă vreme defectele fizice pot fi efectiv reprezentate prin defecte logice, nu acelaşi lucru se întâmplă cu defectele de proiectare. De aceea, în abordarea discuţiei diagnosticului şi reparaţiei subiectul se va restrânge doar la defectele fizice şi logice.

Sunt posibile două tipuri de abordări pentru diagnosticul defectelor. Prima abordare este *analiza cauză-efect,* care enumeră toate defectele (cauzele) posibile să existe într-un model al defectului şi determină, înainte de experimentul de testare, toate răspunsurile (efectele) corespunzătoare unui test aplicat. Acest proces care se bazează pe simularea defectelor, construieşte o bază de date numită *dicţionarul defectelor.*

Diagnosticul este un proces de căutare în dicţionar, în care se încearcă să se potrivească răspunsul actual al UAT cu unul din răspunsurile precalculate. Dacă potrivirea are loc, dicţionarul de defecte indică defectele posibile (şi/sau componentele defecte) din UAT.

Alte tehnici de diagnostic, cum ar fi testarea cu proba ghidată, folosesc o abordare de tipul *analiza efect-cauză.* O analiză efect-cauză procesează răspunsul actual al UAT (efectul) şi încearcă să determine direct numai defectele (cauzele) care ar putea produce respectivul răspuns.

Generarea Testului

Generarea Testului, (GT) este procesul de determinare a stimulilor necesari să testeze un sistem numeric. GT depinde în primul rând de metoda de test folosită.
Metodele de testare "*online*" nu necesită GT. Un mic efort este necesitat pentru GT atunci când vectorii de intrare (*de test*) sunt furnizaţi de un registru de deplasare cu reacţie lucrând ca un generator de secvenţe pseudoaleatoare.
În schimb, GT pentru testarea şi verificarea proiectării, ca şi pentru dezvoltarea programelor de diagnostic implică, în general, un efort considerabil care din nefericire este încă în mare măsură o activitate manuală.

Generarea automata a testului (GTA) se referă la algoritmi care, dat fiind un model al sistemului, pot genera teste pentru acesta. GTA au fost dezvoltate în mod deosebit pentru testarea la nivel de pini-conectori cu secvenţe de vectori de test.

GT poate fi orientată pe *defecte* sau orientată pe *funcţie*. În GT orientată pe defecte, GT încearcă să genereze testele ce vor detecta (si posibil localiza, în anumite limite de distinctibilitate) defectele specificate.

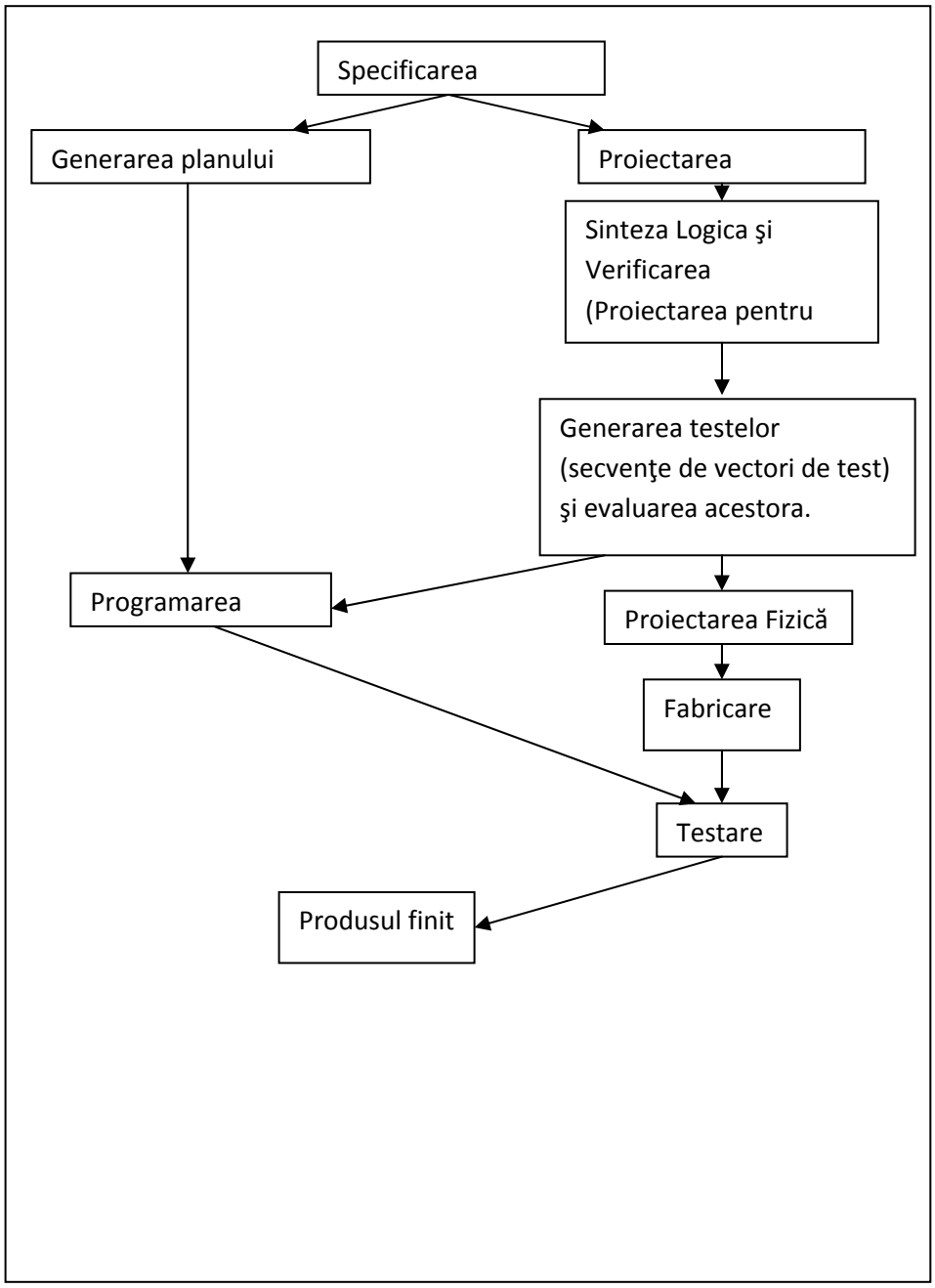
Proiectare pentru Testabilitate

Costul unei testări pentru un sistem poate deveni o componentă majoră în costul proiectării, fabricaţiei şi întreţinerii acestuia. Costul testării reflectă mulţi factori, cum ar fi costul GT, costul timpului de testare, costul echipamentului automat de testare (costul EAT) etc.

Este cumva uimitor faptul că un microprocesor cu un cost de circa 10 USD, necesită testere de sute de ori mai scumpe.

Proiectarea pentru testabilitate (PPT) cumulează o seamă de algoritmi şi tehnici care s-au dezvoltat mult în ultimul timp. Scopul acestora este reducerea costului testării prin introducerea criteriilor de testabilitate cât mai devreme cu putinţa în etapa de proiectare.

Consideraţiile de testabilitate au devenit atât de importante încât au ajuns chiar să impună structură generală a unui proiect.


Specificarea
Generarea planului
Proiectarea
Sinteza Logica şi
Verificarea
(Proiectarea pentru
Generarea testelor
(secvenţe de vectori de test)
şi evaluarea acestora.
Programarea
Proiectarea Fizică
Fabricare
Testare
Produsul finit