UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Instrumente Structurale
2007-2013

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

Testarea Sistemelor

22. Arhitecturi specifice BIST

Arhitecturi specifice BIST

În această secţiune vor fi descrise câteva arhitecturi BIST propuse de cercetători şi grupuri de dezvoltare hardware.

1. Arhitectura *Centralized and Separate Board-Level BIST Architecture* (CSBL)

Arhitectura CSBL a fost introdusă în 1975 de Benowitz şi ceilalţi şi este prezentată în figura 1.

Principalele atribute ale acestei arhitecturi sunt:
Arhitectură BIST centralizată şi separată,
Nu necesită facilităţi *Boundary Scan*,
Circuitul testat poate fi combinaţional ori secvenţial.

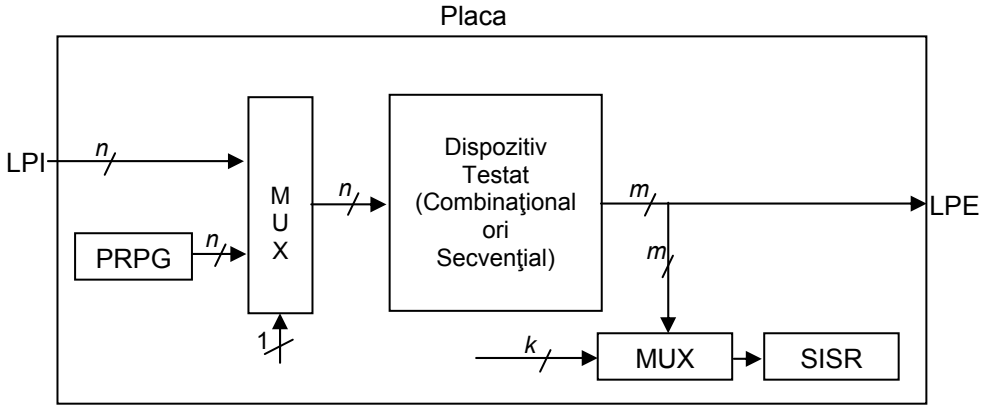

Figura 1. Arhitectura CSBL.

Liniile primare de intrare (LPI), pe durata modului de testare off-line, sunt conduse printr-un dispozitiv *PRPG* (un generator de vectori generaţi pseudo-aleator) iar liniile primare de ieşire (LPI) sunt monitorizate prin intermediul unui dispozitiv *SISR* (analizator de semnătură logică cu intrare unică).
Numărul de linii de selecţie ale multiplexorului, k, are expresia $k = 1 + E(\log_2 m)$, unde $E()$ este funcţia partea întreagă.
S-a ales un dispozitiv SISR în vederea reducerii costurilor circuitelor, testele sunt repetate de m ori, odată pentru fiecare linie primară de ieşire.

Această arhitectură este cel mai potrivită pentru structurile în bandă de asamblare (*pipeline*) cu un volum limitat de conexiuni reactive ieşire-intrare.
Controlerele implementate prin structuri bazate maşini cu stări finite pot prezenta complexitate ridicată de testare.
Microprocesoarele prezintă o situaţie de test deosebită ca şi complexitate prin această arhitectură.
Arhitectura CSBL necesită un efort extins de simulare în vederea stabilirii numărului de vectori de test astfel încât să se atingă un nivel adecvat al acoperirii defectelor.

2. Arhitectura *Built-In Evaluation and Self-Test* (BEST)

Această arhitectură este o aplicaţie a arhitecturii CSBL proiectată pentru circuitele integrate, aşa cum se poate vedea în figura 2.
Logica testată prin această arhitectură este de natură secvenţială.
Liniile de intrare în circuitul testat (CUT) sunt conduse printr-un PRPG iar liniile de ieşire sunt comprimate printr-un MISR.

Detaliile comutării între liniile primare de intrare şi ieşirea dispozitivului PRPG atunci când sunt aplicate CUT, în mod normal ori în regim de test, nu sunt enunţate.
Fie se utilizează un multiplexor, fie valorile liniilor de intrare primare pot fi întâi încărcate în PRPG şi apoi aplicate CUT.

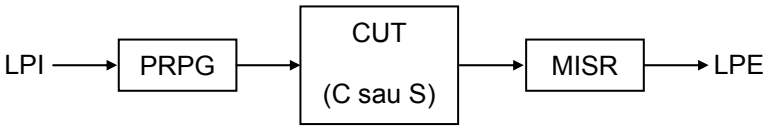

Figura 2. Arhitectura BEST.

Aceleaşi concepte se aplică şi liniilor primare de ieşire. Există atât o versiune dedicată (*embedded*) cât şi una separată a acestei arhitecturi.

Dacă liniile primare de intrare ale CUT trec direct în registre iar liniile primare de ieşire sunt conduse prin registre atunci se poate utiliza implementarea dedicată (*embedded*) a acestei arhitecturi BIST.
Aceasta revine la a spune că aceste registre I/E pot fi modificate pentru o utilizare ca dispozitive PRPG şi MISR.

Altfel, PRPG şi MISR trebuie să fie adăugate la CUT ceea ce conduce la o arhitectură separată. În ultima variantă, dispozitive PRPG şi MISR pot fi făcute să fie parte a registrelor *boundary-scan*.
Volumul de logică implementată suplimentar, pentru această arhitectură BEST, este redus.
Ca şi la arhitectura precedentă este necesară o simulare extensivă a defectelor pentru determinarea unui echilibru acceptabil între acoperirea defectelor şi lungimea testelor.
Pentru anumite circuite această tehnică poate fi ineficientă în ceea ce priveşte atingerea unui nivel acceptabil de acoperire al defectelor.

3. Arhitectura *Random-Test Socket* (RTS)

Arhitectura RTS a fost descrisă de Bardell şi McAnney în 1982. Aşa cum au precizat autorii acestei arhitecturi, această arhitectură nu este în totalitate, stricto-senso, o arhitectură BIST deoarece circuitele, în întregime, care efectuează testul sunt externe circuitului testat (CUT).

Arhitectura RTS prezentată schematic, în figura 3, are următoarele atribute fundamentale:

- Circuitele de teste sunt distribuite şi separate,
- Nu are implementată disciplina *boundary scan*,
- Arhitectura de scan a căilor dispozitivului testat este LSSD.

Testarea are loc astfel:

1. Se iniţializează registrele LSFR,
2. Se plasează un vector de test, generat pseudo-aleatoriu, în calea de scan prin încărcarea registrului R_2.
3. Se generează pseudo-aleatoriu un nou vector de test utilizând registrul R_1. Aceasta are loc pe durata unui impuls de ceas.
   Vectorul este aplicat liniilor primare de intrare ale dispozitivului testat (CUT).
4. Se culege răspunsul dispozitivului testat (CUT) la nivelul liniilor primare de ieşire dispozitivului testat prin aplicarea unui impuls de ceas registrului R_3.
5. Se execută o operaţie de încărcare paralelă în celule de stocare ale sistemului pentru capturarea răspunsului după aplicarea vectorului de test generat aleatoriu.
6. Se scanează spre exterior datele din calea de scan a dispozitivului testat (CUT) şi se comprimă aceste date în registrul R_4.

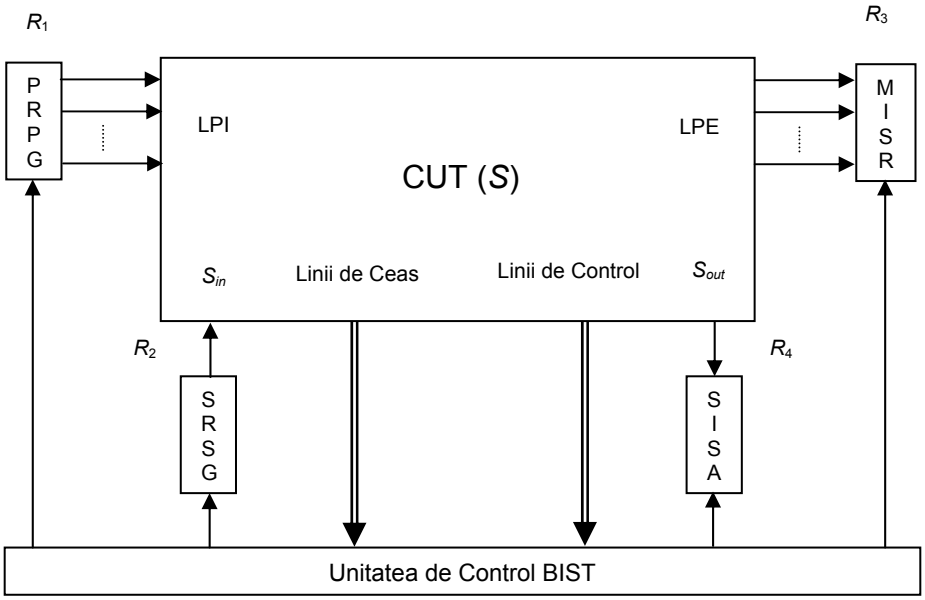

Figura 3. Arhitectura BIST *Random-Test Socket* (RTS).

Etapele, paşii din procedeul mai sus enunţat între 2 şi 6, sunt repetate până când se atinge un nivel adecvat de acoperire al defectelor ori până când s-a scurs timpul maxim disponibil de testare. Este de remarcat faptul că etapele între 2 şi 6 pot fi desfăşurate simultan.
Dispozitivului testat (CUT) se consideră că funcţionează corect atunci când valorile finale aflate în registrele R_3 şi R_4 sunt corecte.

Abordarea testării prin arhitectura RTS poate suferi limitări legate de timpul alocat procesului de testare dar şi din raţiuni relative la acoperirea defectelor.
Testarea este inerent lentă, larg consumatoare de timp, deoarece pentru fiecare vector de test generat aleatoriu trebuie să se încarce în întregime calea de scan.

Timpul dedicat testării poate fi micşorat, chiar redus substanţial, prin partiţionarea celulelor de stocare în câteva căi de scan.
Dar, această soluţie necesită introducere unor pini de intrare / ieşire suplimentari actualei arhitecturi.

Testarea cu vectori de test generaţi aleatoriu impune utilizare unui număr mai mare de vectori de test comparativ cu testare prin vectori de test generaţi deterministic.

Volumul de vectori de test este determinat de nivelul impus de acoperire al defectelor, de circuitul propriu-zis dar şi de caracteristicile circuitelor LFSR utilizate pentru generarea datelor de test şi pentru producerea semnăturilor.

Un nivel ridicat de acoperire al defectelor se soluţionează prin creşterea corespunzătoare a numărului de teste.
O soluţie eficientă care să menţină în volume mai mici testele este introducerea punctelor de test.

Autorii acestei arhitecturi, Bardell şi McAnney, dar şi Chen, separat, au arătat în 1986 că acoperirea defectelor poate fi negativ influenţată de corelaţiile liniare dintre biţii generaţi de registrele LFSR şi periodicitatea secvenţelor generate de aceste registre.
Astfel, se presupune că registrul LFSR R_2 are perioada p iar lungimea căii de scan este k şi fie k divide p, fie p divide k.
Atunci, în primul caz, se poate demonstra că doar k/p vectori unici pot fi încărcaţi în calea de scan.
Pe de-altă parte, în al cel de-al doilea caz, se dovedeşte că există doar p/k vectori de test distincţi.
Din aceste raţiuni este deosebit de important ca niciuna dintre cele două situaţii să nu aibă loc.
De asemenea, dacă nu se implementează un polinom primitiv atunci nu sunt generaţi sunt toţi vectorii de test.
Există posibilitatea ca unii dintre aceşti vectori de test, cei negeneraţi, să fie necesari pentru detecţia unor defecte specifice.
Chiar şi atunci când $p >> k$, o încărcare completă a căii de scan între fiecare vector de test implică faptul că largi blocuri de logică combinaţională din dispozitivul testat (CUT) nu vor fi exhaustiv testate.

4. Arhitectura *Built-In Logic-Block Observation* (BILBO)

O problemă majoră cu anumite arhitecturi BIST constă în faptul că acestea abordează o versiune nepartiţionată a dispozitivului aflat sub test (CUT).

Aceasta este echivalent cu a remarca faptul că toate liniile de intrare primare sunt grupate laolaltă într-o mulţime, toate liniile primare de ieşire sunt grupate într-o a doua mulţime iar celule de memorie într-o a treia mulţime.

Aceste seturi sunt asociate cu registrele PRPG şi MISR. Deoarece numărul de ranguri din aceste registre sunt, de regulă, mari nu este fezabilă abordarea prin tehnici exhaustive ori pseudo-exhaustive.
Un circuit integrat de interes poate avea, în mod curent, peste 100 de pini de intrare şi câteva sute de celule de memorie.

O abordare a acestei probleme poate fi considerată prin dispunerea celulelor de memorie în grupuri, numite de regulă registre.
Aceste grupuri corespund unor registre funcţionale care se pot regăsi în multe proiecte.
Astfel, se pot întâlni numărătorul de instrucţiuni program, registrul instrucţiunii, acumulatorul şi altele.

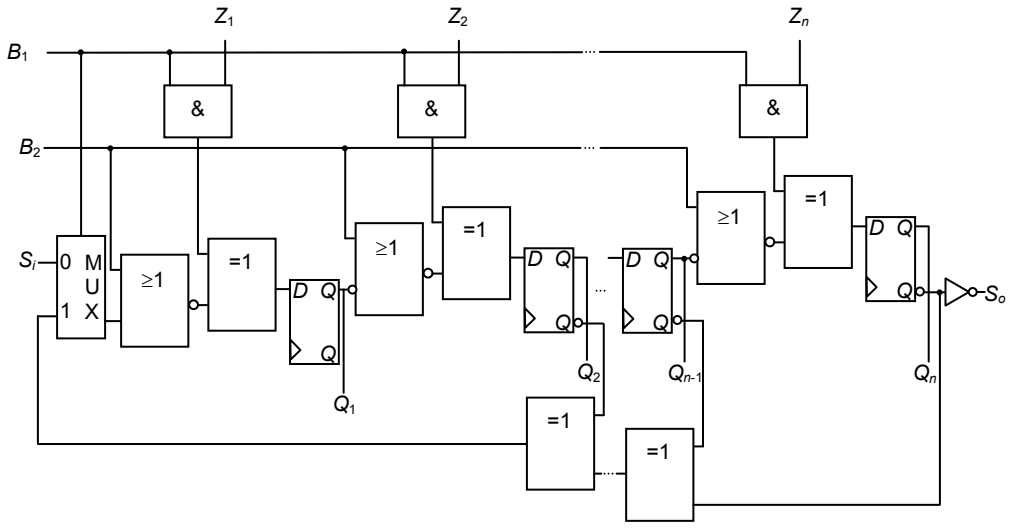

Figura 4.a. Registrul BILBO cu *n* biţi.

Arhitectura BILBO, introdusă de Koenemann şi colaboratorii săi în anii 1979 şi 1980, utilizează aceste aspecte ale registrelor regăsibile în multe circuite integrate aflate curent în producţie (aşa cum se poate vedea în figura 4.a).

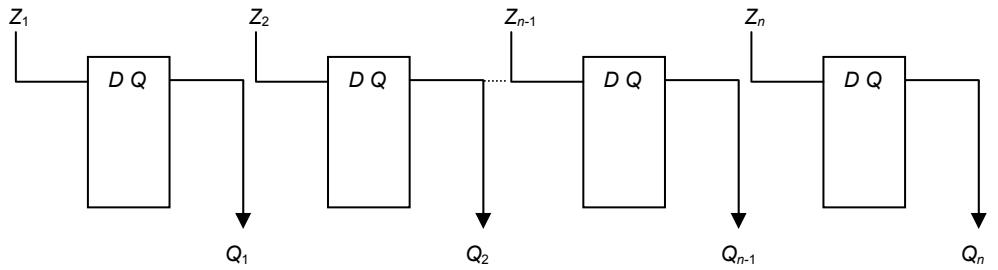

Figura 4.b. Modul normal
de încărcare paralelă ($B_1 = B_2 = 1$).

În această arhitectură a registrelor linia de ieşire complementată *Q'* a unei celule de memorie este conectă printr-o suită de porţi NOR şi XOR la intrarea de date a următoarei celule de memorie.
Un registru de memorie operează într-unul din cele patru moduri posibile, aşa cum este specificat de liniile de control B_1 şi B_2. Atunci când $B_1 = B_2 = 1$, registrul BILBO operează în modul normal de încărcare paralelă date (figura 4.b).

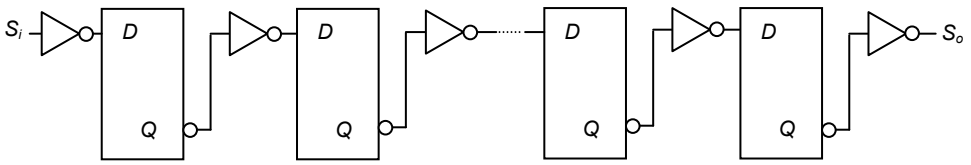

Figura 4.c. Modul de funcţionare registru de deplasare
($B_1 = B_2 = 0$).

Iar atunci când $B_1 = B_2 = 0$, registrul BILBO operează în modul registru de deplasare cu intrarea de *scan* S_i (figura 4.c).

De remarcat faptul că datele sunt complementate atunci când intră în registrul de scan. Atunci când $B_1 = 0$ şi $B_2 = 1$, toate celule de memorie sunt resetate.
Pentru situaţia când $B_1 = 1$ şi $B_2 = 0$, registrul BILBO este configurat să funcţioneze ca şi LFSR (figura 4.d) sau, mai precis, să opereze ca un registru MISR.

Dacă Z_i sunt liniile de ieşire din dispozitivul verificat (CUT), atunci registrul comprimă răspunsul şi generează o semnătură.
Iar dacă intrările $Z_1, Z_2, \ldots, Z_n$, sunt menţinute constant la valoarea 0 şi valoarea iniţială a registrului nu este identic zero, atunci LFSR operează ca un generator de vectori pseudo-aleatori.

Arhitectura de bază BILBO constă din partiţionarea circuitului testat într-un set de registre şi blocuri de logică combinaţională, în care registrele normale sunt înlocuite prin registre BILBO.
Suplimentar, intrările într-un bloc logic *C* sunt conduse printr-un registru BILBO R_i iar ieşirile circuitului combinaţional *C* pilotează un alt registru R_j.

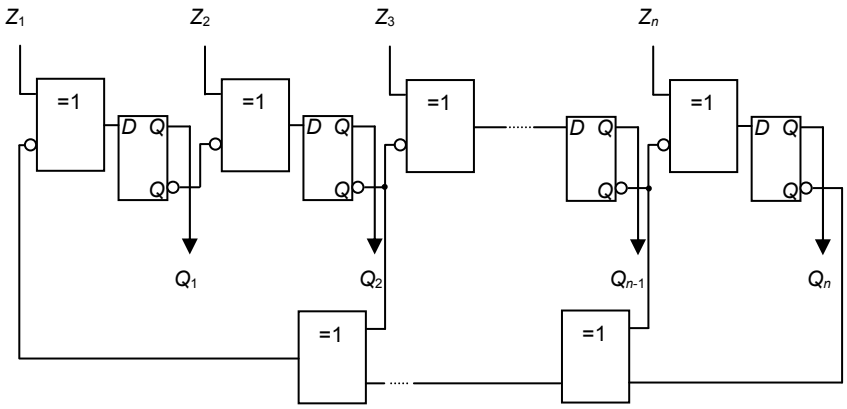

Figura 4.d. Modul de funcţionare LFSR (test)
(B_1 = 1, B_2 = 0).

Se consideră circuitul din figura 5, în care toate registrele sunt *BILBO*-uri. Pentru testarea blocului combinaţional $C1$, sunt întâi încărcate registrele R_1 şi R_2 cu „sămânţă" iar apoi registrul R_1 este poziţionat în modul PRPG iar registrul R_2 este poziţionat în modul MISR.

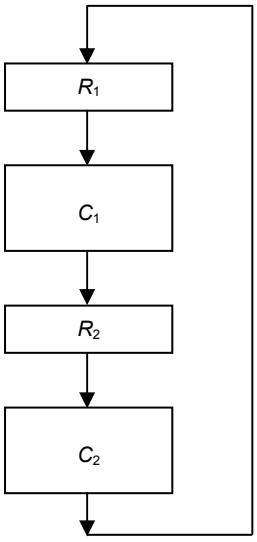

Figura 5. Proiectare BIST cu registre BILBO.

Se presupune că liniile de intrare în registrul R_1 sunt menţinute în valoarea 0. Circuitul este rulat în acest mod pe durat a N cicluri de ceas.
Dacă numărul de linii de intrare în circuitul C_1 nu este prea mare, atunci circuitul C_1 este posibil să fie testat exhaustiv (exceptând vectorul cu toate componentele nule).
La sfârşitul acestui proces, numit sesiune de test, conţinutul registrului R_2 poate fi scanat în afară şi, astfel, verificată semnătura.
În mod analog, se poate testa circuitul combinaţional $C2$, prin configurarea registrului R_1 ca MISR iar registrul R_2 ca PRPG.
Astfel, întreg circuitul este testat în două sesiuni.