

Arhitecturi Built-In Self-Test

Probleme propuse

Oprîtoiu Flavius
flavius.opritoiu@cs.upt.ro

22 noiembrie 2024

Introducere

Obiective:

- ▶ Configurarea unei arhitecturi BIST

Erorile sunt definite în raport cu serviciile oferit de un sistem [ALRH04]. Serviciile unui sistem reprezintă o succesiune de stări externe iar o eroare apare atunci când cel puțin una din stările externe deviază față de comportamentul corect [ALRH04].

Un *defect* reprezintă cauza ipotetică a unei erori iar *toleranța la defectare* oferă mijloace de atingere a dependibilității și securității în sisteme de calcul prin evitarea eșecurilor sistemelor în prezența defectelor [ALRH04].

Detecția erorilor afectând circuite combinaționale

Tehnicile de *detecție a erori* identifică prezența erorilor și sunt clasificate în:

- tehnici **concurrente**, sau
- tehnici de tip **pre-emptive**

Mecanismele de detecție concurrentă operează pe durata funcționării normale a sistemului iar cele pre-emptive suspendă funcționarea normală și aduc sistemul într-un mod de test [ALRH04].

Extinderea unui sistem prin tehnici de testare concurrentă adaugă sistemului capabilități de *auto-testare*, sistemul putându-și verifica funcționarea corectă.

Detecția concurență a erorii

Metodele de detecție concurență a erorii includ:

- duplicare hardware,
- redundanță de cod,
- redundanță temporală

Duplicarea hardware adaugă o copie, modulului care trebuie protejat, comparând continuu ieșirile celor 2 copii. Diferențele între cele 2 ieșiri semnaleză prezența unei erori.

Redundanța de cod verifică faptul că ieșirea păstrează o așa-numită *proprietate de tip invariant*. *Exemplu:* Pentru o unitate care primește 2 valori fără semn, multipli de 3, și calculează suma lor, un invariant (considerând că nu apare overflow) este faptul că rezultatul va fi un multiplu de 3.

Detecția pre-emptivă a erorii

Metodele de detecție a erorii, de tip pre-emptiv, suspendă funcționarea normală a sistemului aducându-l într-un mod de funcționare pentru testare [ALRH04]. Două mecanisme, utilizate în mod curent, se disting în această categorie:

- **testare de tip scan**, and
- **Built-In Self-Test (BIST)**

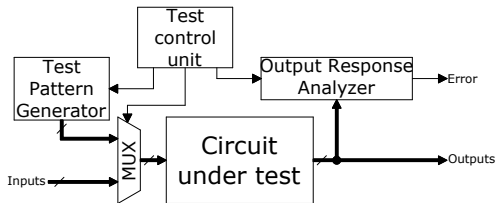
Testarea de tip scan modifică un dispozitiv secvențial prin conectarea serială, a tuturor/majorității elementelor de stocare, într-un **canale de scanare** dedicate, care nu sunt altceva decât registre de deplasare cu acces la exterior.

Metodele BIST transformă un design într-o arhitectură cu facilități de auto-testare, capabilă să detecteze prezența erorilor într-o manieră autonomă, aspect care recomandă arhitectura pentru sisteme cu cerințe de siguranță în funcționare.

BIST

Metoda BIST de detecție a erorilor transformă un design într-o arhitectură auto-testabilă, capabilă să detecteze prezența erorilor în mod autonom.

O arhitectură BIST tipică este descrisă mai jos:



Test Pattern Generator (TPG) generează vectori de test, care vor fi conectați la intrările unității Circuit Under Test (CUT). Output Response Analyzer (ORA) analizează rezultatele CUT-ului pentru detectarea erorilor. În cazul unui CUT combinațional, pentru fiecare vector de test aplicat, se obține un vector răspuns la ieșirile CUT-ului.

Unitatea TPG

Unitatea TPG poate fi construită utilizând:

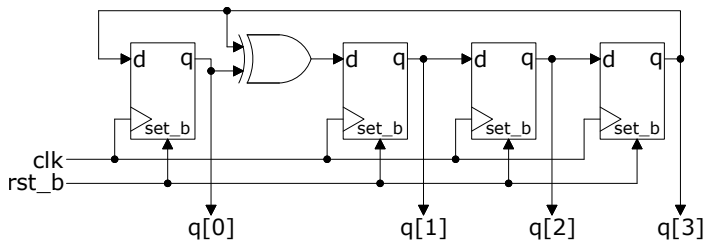
- numărătoare binare, sau
- Linear Feedback Shift Registers (LFSRs)

Numărătoarele binare generează toate configurațiile de intrare ale CUT-ului, exhaustiv.

LFSR reprezintă mecanismul convențional de generare a vectorilor de test în structurile BIST. Sunt construite ca registre de deplasare cu o conexiune de reacție, prelucrată prin porți EXOR.

LFSR-uri

Figura de mai jos ilustrează o structură LFSR pe 4 ranguri:




















Când este inițializat cu un vector ne-nul, un LFSR generează, la ieșire, o secvență pseudo-aleatorie, repetitivă.

Pentru arhitectura de mai sus, secvența de ieșire, compusă din vectori pe 4 biți, se repetă cu o periodicitate de 15 (sunt generați toți vectori ne-nuli pe 4 biți).

LFSR-uri (contin.)

Cei 15 vectori, generați la ieșirea LFSR-ului de mai sus, sunt:

rst_b	clk	q[3]	q[2]	q[1]	q[0]
0	<i>d</i>	1	1	1	1
1		1	1	0	1
1		1	0	0	1
1		0	0	0	1
1		0	0	1	0
1		0	1	0	0
1		1	0	0	0
1		0	0	1	1
1		0	1	1	0
1		1	1	0	0
1		1	0	1	1
1		1	0	1	1
1		0	1	0	1
1		1	0	1	0
1		0	1	1	1
1		1	1	1	0
1		1	1	1	1
1		1	1	0	1

↑ Output sequence periodicity ↓

Unitatea ORA

ORA efectuează compactarea datelor (cu pierdere de informație) procesând toate rezultatele CUT-ului atunci când acesta este exersat cu vectorii de test generați de TPG. La finele compactării, ORA furnizează o *semnătură*. Semnătura este un vector, restrâns, de lungime fixă, care caracterizează întregul set de rezultate.

Semnătura unui CUT este asociată cu unitatea TPG care generează intrările pentru CUT. *Semnătura de aur* se referă la semnătura obținută pentru un CUT neafectat de defecte. De regulă, este obținută prin simulare.

Prezența erorilor într-un CUT este detectată prin comparația semnăturii obținute cu semnătura de aur.

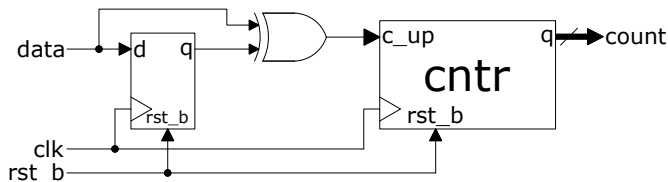
Unitatea ORA poate fi implementată utilizând:

- thenici de numărare, sau
- registre de semnătură

Tehnici de numărare

Tehnicile de numărare pot cuantifica fie numărul de apariții ale unei valori logice la o ieșire, fie numărul de tranziții ale unei linii de ieșire. Pentru numărarea unei valori logice (1 sau 0) la o ieșire se folosesc numărătoare binare.

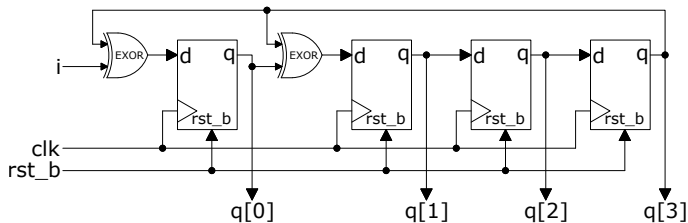
Un numărător de tranziții este ilustrat mai jos:



O unitate de numărare se va conecta la fiecare linie a ieșirii CUT-ului. În consecință, o ieșire de 4 biți necesită 4 instanțe numărător. Pentru o singură linie, semnătura finală este reprezentată de conținutul numărătorului după primirea tuturor biților acelei linii.

Registre de semnătură

Un **Single Input Signature Register (SISR)** este construit în jurul unui LFSR având o intrare de date, adițională. SISR-ul construit pornind de la arhitectura LFSR prezentată anterior este ilustrat mai jos:



SISR-ul necesită inițializarea cu configurația zero.

O unitate SISR va fi conectată la fiecare linie a ieșirii unui CUT iar semnătura finală reprezintă conținutul SISR-ului după procesarea tuturor biților recepționați.

Referințe

- [ALRH04] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, “Basic Concepts and Taxonomy of Dependable and Secure Computing,” *IEEE Trans. Dependable Secur. Comput.*, vol. 1, no. 1, pp. 11–33, Jan. 2004.