

S10. Metode de testare hardware

Arhitecturi de autotestare implementate în Verilog

1. Scopul laboratorului

Configurarea si implementarea în Verilog a unei arhitecturi BIST

2. Introducere

Erorile sunt definite în raport cu serviciile oferite 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 suferă o abatere față de comportamentul corect [ALRH04].

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

3. Desfășurarea laboratorului

3.1 BIST

Metoda Built-In-Self-Test (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 poate fi vizualizată pe pagina următoare (vezi **Fig. 1**). După o analiză sumară a arhitecturii vom putea identifica următoarele elemente:

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

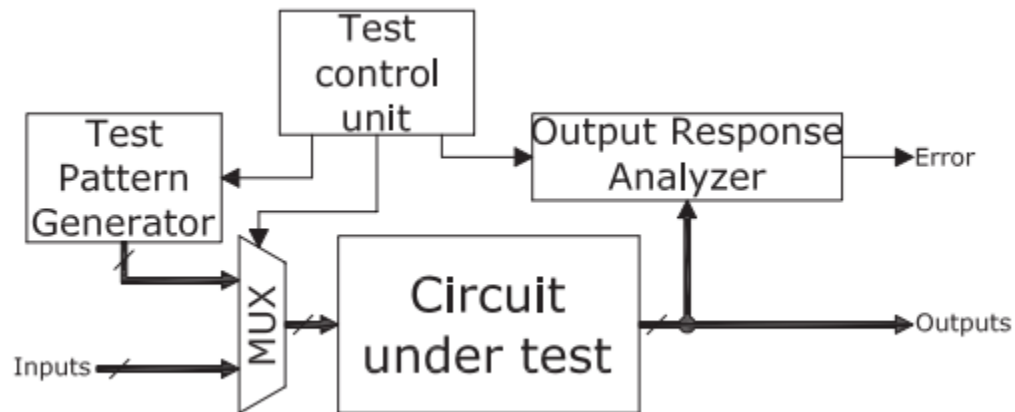


Fig.1 – Built-In-Self-Test Architecture

3.2 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 EX-OR.

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

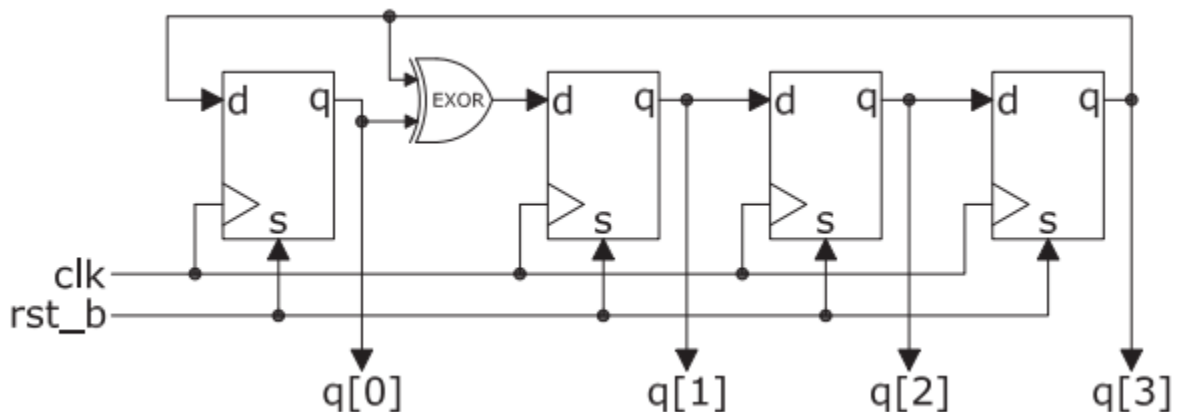

















Fig.2 – LFSR with EXOR feedback path

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

Pentru arhitectura din **Fig. 2**, secvența de ieșire, compusă din vectori pe 4 biți, se repetă cu o periodicitate de 15 (sunt generați toți vectorii ne-nuli pe 4 biți).

Cei 15 vectori, generați la ieșirea LFSR-ului dat, 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		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		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

Periodicitatea secvenței de ieșire

La început, un LFSR este definit ca un vector inițializat pe 1 pe fiecare poziție, pentru valoarea **rst_b** = 0 și la orice semnal de clock (simbolizat *d* sau *). La frontul crescător a semnalului de tact, când **rst_b**=1 se produce prima schimbare în vectorul de biți, **q[3]** preia vechea valoare a lui **q[2]**, **q[2]** ia valoarea lui **q[1]**. La acest punct $q[1] = q[0] \oplus q[3]$ (vezi **Fig.2**), adică **q[1]**=0. În final **q[0]** ia vechea valoare a lui **q[3]**.

Pentru următoarea iterație se reia procedura de mai sus, generând astfel un număr de 15 vectori. Periodicitatea secvenței de ieșire este sesizabilă la cea de-a 15-a iterație menționată în tabel.

3.3 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:

- tehnici de numărare, sau
- registre de semnătură

3.4 Tehnici de numărare

Tehnicile de numărare pot cunatifica 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 (0 sau 1) la o ieșire se folosesc numărătoare binare.

Un numărător de tranziții este ilustrat mai jos:

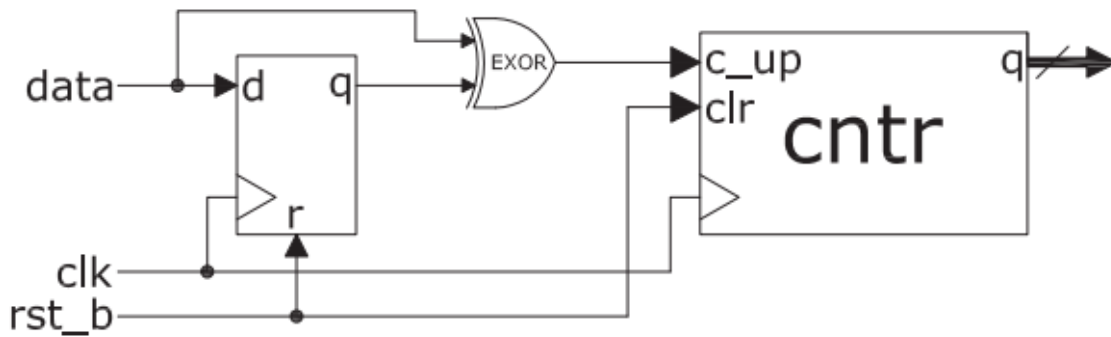


Fig.3 – Transition Counter connected to a Register

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

3.5 Registre de semnătură

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

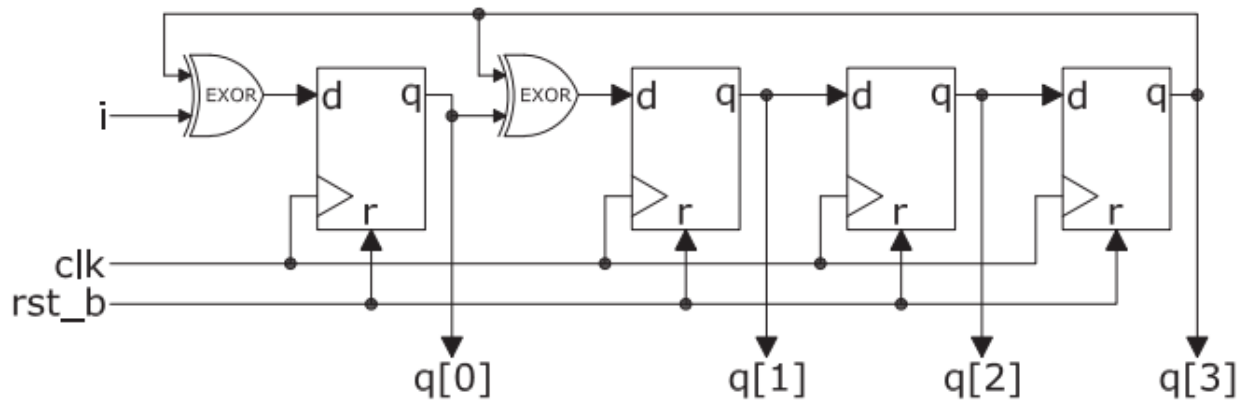




Fig.4 – Single Input Signature Register Architecture

SISR-ul, față de LFSR, pornește de la inițializarea vectorului de biți cu valoarea 0 pe toate cele 4 poziții (SISR-ul fiind de rang 4).

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.

4. Probleme propuse

Problema 1. Considerând tabelul completat de la pagina 3 pentru **LFSR**, se vor genera 15 vectori de biți pentru **SISR**, pornind de la capul de tabel prezentat mai jos:

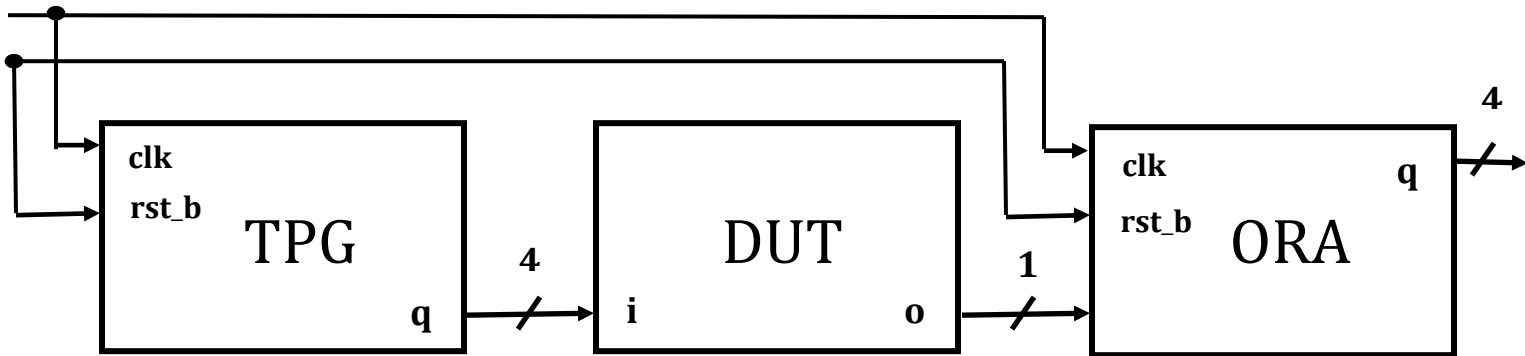
rst_b	i	clk	q[3]	q[2]	q[1]	q[0]
0	d	d	0	0	0	0
1	1	
1	1	

Vectorul de intrare **i** va primi următoarea secvență de biți [1 1 0 1 1 0 1 0 0 1 1 1 1 0 0]. Biții vor fi trecuți în tabel pe coloana lui i de la stânga la dreapta primul bit 1 → ultimul bit 0.

Cerință: Să se completeze tabelul construit până la ultima iterație și să se determine vectorul semnătură a SISR-ului.

Problema 2. Să se implementeze, folosind limbajul Verilog, arhitectura **SISR** prezentată în **Fig. 4**.

Problema 3. Să se realizeze, folosind limbajul Verilog, următoarea arhitectură **BIST** (Built-In-Self-Test):



Pentru arhitectura dată se cunosc următoarele elemente:

- Test Pattern Generator (TPG) – este un numărător binar pe 4 biți.
- Device Under Test (DUT) – este un circuit pur combinațional la care ieșirea **o** a blocului va fi **0** dacă intrarea **i** este multiplu de 3 și **1** în caz contrar.
- Output Response Analyzer (ORA) – blocul poate să fie substituit fie de un **SISR** sau un **numărător de tranziții**.

Să se determine **semnătura de aur** a ieșirii prin simularea arhitecturii implementate.

5. Referințe bibliografice

[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.