Built-In Self-Testing

M. Favalli

Engineering Department in Ferrara

Introduzione

Sommario

Motivazioni - I

- Collaudo din un sistema, scheda o circuito integrato durante la loro normale vita operativa
- Applicato inizialmente a livello di sistema con approcci basati su software specifico per collaudo e diagnosi che risultavano in
  - bassi livelli di copertura
  - bassa risoluzione nella diagnosi
  - scarsa velocitá
- Il BIST hardware a livello di sistema
  - semplifica il collaudo e la diagnosi
  - migliora l'affidabilitá del sistema
  - migliore mantenimento del sistems

Motivazioni - II

- Il BIST é stato poi esteso al livello di circuito integrato
- Impatto sul collaudo di produzione
  - migliora controllabilitá e osservabilitá
  - semplifica i compiti della test generation e riduce il tempo di applicazione delle sequenze di collaudo
  - riduce la lunghezza delle sequenze di collaudo degli ATE
  - consente ad ATE lenti di collaudare i circuiti alla loro normale velocitá di funzionamento
- Consente di collaudare il circuito durante la sua vita operativa migliorando l'affidabilitá dei sistemi in cui é inserito

Idee base del BIST

- Si cerca di portare l'ATE sul circuito integrato
- Il numero elevato di vettori di test gestiti dall'ATE richiederebbe di inserire nell'integrato una memoria di grandi dimensioni per contenere i vettori di ingresso e uscita
- Bisogna trovare il modo di generare i vettori di ingresso con circuiti compatti verificare la correttezza della risposta con circuiti compatti

Costi

- Area overhead
  - test controller
  - hardware pattern generator
  - hardware response analyzer
  - collaudo dell'hardware di BIST
- Pin overhead, almeno un pin
- Overhead sulle prestazioni
  - ritardi addizionali
  - incremento nel consumo di potenza
- Perdita di resa dovuta all'incremento di area
- Riduzione dell'affidabilitá sempre dovuta all'incremento di area

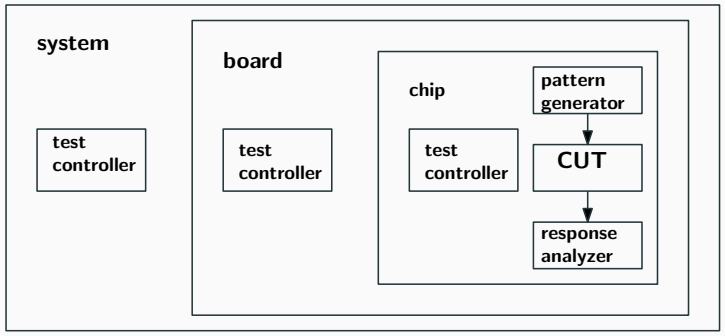
Guasti da collaudare da parte del BIST

Per il BIST si possono definire obbiettivi di collaudo per il collaudo di produzione (dove il BIST lavora con un ATE) e per il collaudo durante la vita operativo del circuito

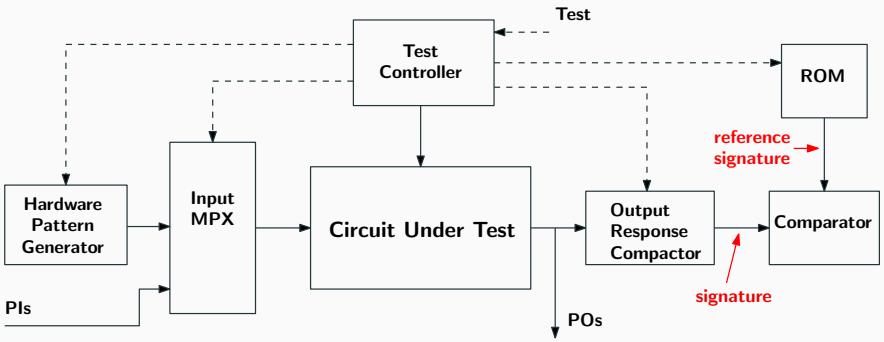
- stuck-at singoli
- delay faults
- transistor faults
- bridgings
- stuck-at nell'hw di BIST

BIST - test controller

- Hardware dedicato che attiva i processi di collaudo a livello di sistema, scheda e circuito integrato
- Queste operazioni possono essere svolte in parallelo



Architettura di BIST



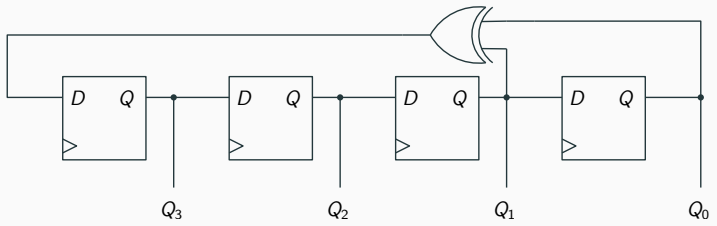
- Il BIST non puó collaudare interconnessioni e transistor dai PI fino alla logica di ingresso al MPX e dall'uscita del CUT fino ai PO

Hardware pattern generator

- Traduzione diretta di un test set deterministico in hardware: ROM e contatore per generare gli indirizzi
  - area eccessiva
- Collaudo esaustivo: un contatore genera tutti i possibili vettori di test
  - tempi di collaudo eccessivi
  - se si riduce il numero di vettori di test la loro distribuzione risulta tipicamente inadatta al collaudo
- Collaudo pseudocasuale: viene generata una sequenza pseudocasuale di lunghezza inferiore a quella esaustiva mediante un LFSR (o altri tipi di circuito)

Sequenza pseudocasuale

- Sequenza di configurazioni binarie di n bit le cui caratteristiche statistiche sono simili a quelle di una sequenza casuale
- Come conseguenza la probabilitá di avere 1 su uno di tali bit é circa 0.5, quella di avere 01, 10, 11 o 00 su una coppia di tali bit é 0.25 ....
- Possono essere generate con reti compatte come gli Linear Feedback Shift Register (LFSR)



Hardware response analyzer

- Traduzione diretta di un test set deterministico in hardware: ROM per memorizzare le risposte e contatore per generare gli indirizzi
- Collaudo esaustivo e pseudorandom testing
  - compressore: nessuna perdita di informazione, ma area overhead elevato
  - compattatore: perdita di informazione (alcune sequenze di risposte con errori sono classificate come fault-free) con overhead ridotti
- I compattatori possono essere basati su
  - contatori di 1 o di transizioni
  - LFSR con minori probabilitá di produrre falsi negativi

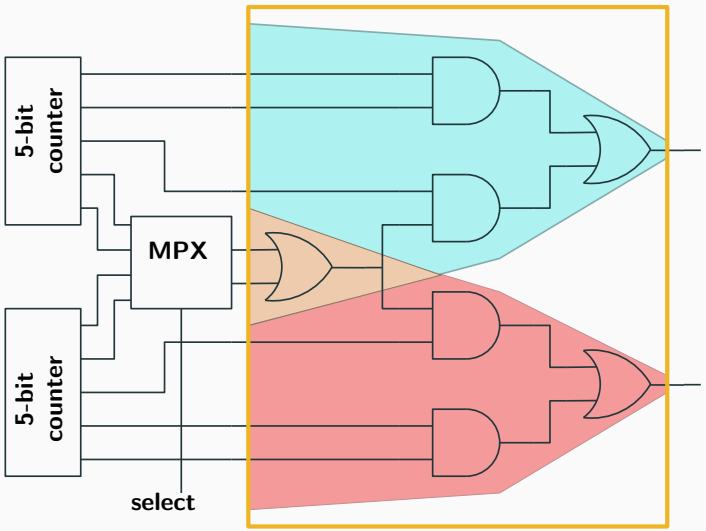
Hardware pattern generator

Test generation pseudoesaustiva

- Si é visto come la generazione esaustiva sia non fattibile
- Si partiziona il circuito in fanin cones ciascuno dei quali viene collaudato esaustivamente
- Backtrace dai PO individuando i PI che li influenzano
- L'operazione non sempre é possibile e la copertura di guasto puó essere incompleta

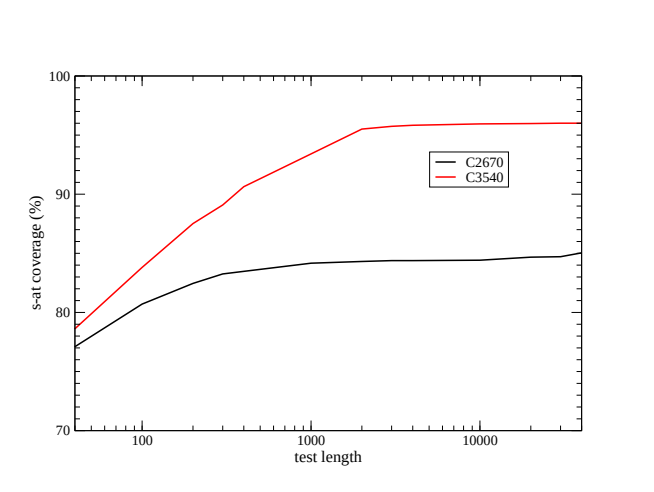
Esempio

Riduzione da 2^8 vettori di test a 2^6



Random pattern testing

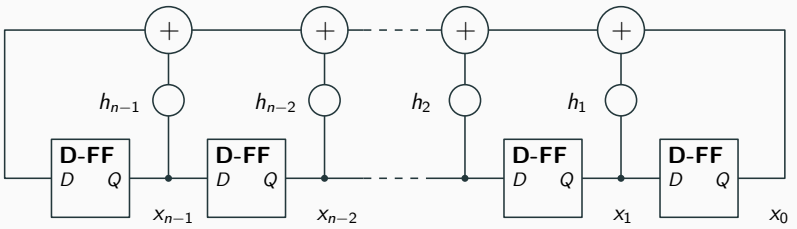
- Il problema di questa tecnica con i circuiti combinatori é data dai guasti random resistant



- I circuiti sequenziali possono dare luogo a ulteriori difficoltá

Pseudo-random testing

- Basato su LFSR o cellular automata
  - producono sequenze deterministiche statisticamente simili a sequenze puramente casuali
  - la lunghezza delle sequenze puó essere determinata sulla base della copertura di guasto attesa
- Rappresentazione parametrica di un LFSR



Modello algebrico

$$\begin{bmatrix} x_0(t+1) \\ x_1(t+1) \\ \cdot \\ \cdot \\ \cdot \\ x_{n-3}(t+1) \\ x_{n-2}(t+1) \\ x_{n-1}(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \ldots. & 0 & 0 \\ 0 & 0 & 1 & \ldots. & 0 & 0 \\ \cdot & \cdot & \cdot & & \cdot & \cdot \\ \cdot & \cdot & \cdot & & \cdot & \cdot \\ \cdot & \cdot & \cdot & & \cdot & \cdot \\ 0 & 0 & 0 & \ldots. & 1 & 0 \\ 0 & 0 & 0 & \ldots. & 0 & 1 \\ 1 & h_1 & h_2 & \ldots. & h_{n-2} & h_{n-1} \end{bmatrix} \begin{bmatrix} x_0(t) \\ x_1(t) \\ \cdot \\ \cdot \\ \cdot \\ x_{n-3}(t) \\ x_{n-2}(t) \\ x_{n-1}(t) \end{bmatrix}$$

$X(t+1) = T_s X(t)$, dove T_s é detta matrice companion

Un LFSR implementa un campo di Galois

- Struttura algebrica
  - moltiplicazione per X realizzata come right shift
  - addizione XOR ($\oplus$)
- Matrice companion
  - nella prima colonna si hanno tutti 0 meno l'n-mo elemento che é 1 perché x_0 alimenta sempre x_{n-1}
  - la riga n-ma rappresenta i coefficenti di retroazione h_i
  - il resto é la matrice identitá I che equivale a un right shift

Funzionamento dell'LFSR

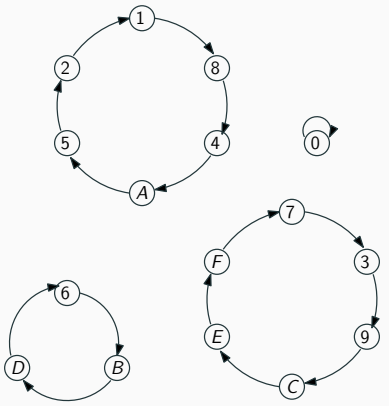
- Non va inizializzato con $X = [0, 0, ...0]^t$
- Se X_0 é lo stato iniziale, lo stato procede come X_0, $T_s X_0$, $T_s^2 X_0$, $T_s^3 X_0$, ...
- Periodo(i) della matrice
  - il piú piccolo k tale che $T_s^k = I$
  - k é la lunghezza di ciclo minima (e diversa da 1) dell'LFSR
- Polinomio caratteristico

$$f(\chi) = |T_s - I\chi| = 1 + h_1\chi + h_2\chi^2 + ... + h_{n-1}\chi^{n-1} + \chi^n$$

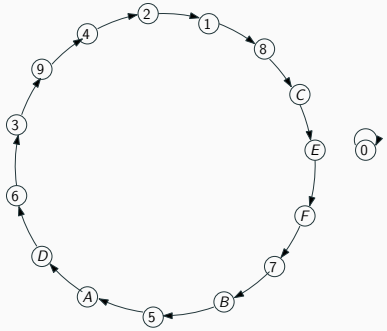
- Se la lunghezza di ciclo é pari a $2^n - 1$ l'LFSR si dice di lunghezza massima

Esempio: STG di due LFSR per $n = 2$

Non maximal lentgh:
$h_1 = 0,\ h_2 = 1,\ h_3 = 0$
$(x^4 + x^2 + 1)$



Maximal length:
$h_1 = 1,\ h_2 = 0,\ h_3 = 0$
$(x^4 + x + 1)$

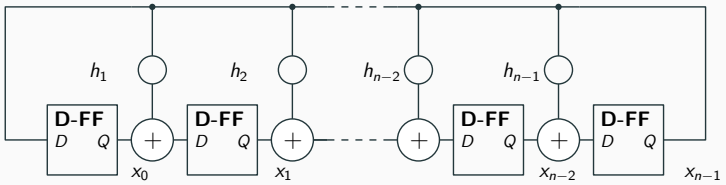


LFSR con lunghezza massima

- Tipicamente si cerca di utilizzare un LFSR con lunghezza massima perché la sequenza prodotta é piú simile a una casuale
- **Polinomio irreducibile**: é un polinomio che non puó essere fattorizzato, il periodo minimo é un divisore di $2^n - 1$
- Un polinomio irreducibile con periodo $2^n - 1$ é un **polinomio primitivo** (é un divisore di $\chi^k + 1$ per $k = 2^n - 1$) ma non per valori piú piccoli)
- Un polinomio primitivo corrisponde a un LFSR di lunghezza massima
- Questi polinomi sono tabulati per diversi valori di n

LFSR interno

- L'LFSR visto fino a questo momento viene detto esterno
- La versione interna ha un funzionamento simile con qualche vantaggio sul ritardo combinatorio



Modello algebrico dell'LFSR interno

- Descritto da $T_m = T_s^t$
- $X(t+1) = T_m X(t)$
- Polinomio caratteristico $f(\chi) = |T_m - IX|$

$$f(\chi) = 1 + h_1\chi + h_2\chi + ... + h_{n-2}\chi^{n-2} + h_{n-1}\chi^{n-1}$$

- Lo shift verso destra equivale a moltiplicare il polinomio rappresentativo di x, poi divide per il polinomio caratteristico e mantiene il resto memorizzato

Modello algebrico dell'LFSR interno

$$\begin{bmatrix} x_0(t+1) \\ x_1(t+1) \\ x_2(t+1) \\ . \\ . \\ . \\ x_{n-2}(t+1) \\ x_{n-1}(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & .... & 0 & 0 & 1 \\ 0 & 1 & 0 & .... & 0 & 0 & h_1 \\ 0 & 0 & 1 & .... & 0 & 0 & h_2 \\ . & . & . & & . & . & . \\ . & . & . & & . & . & . \\ . & . & . & & . & . & . \\ 0 & 0 & 0 & .... & 1 & 0 & h_{n-2} \\ 0 & 0 & 0 & .... & 0 & 1 & h_{n-1} \end{bmatrix} \begin{bmatrix} x_0(t) \\ x_1(t) \\ x_2(t) \\ . \\ . \\ . \\ x_{n-2}(t) \\ x_{n-1}(t) \end{bmatrix}$$

$X(t+1) = T_m X(t)$, dove T_s é detta matrice companion

Tecniche per migliorare un TPG hardware

Come incrementare la copertura di un TPG hardware

- Weighted pseudorandom generation: non é detto che avere una distribuzione uniforme di probabilitá di 1 in ingresso al CUT sia la cosa migliore
  - si puó filtrare l'uscita dell'LFSR con una rete combinatoria in modo da cambiare tali probabilitá (weight)
- Test pattern augmentation: per rivelare i guasti random resistant si puó utilizzare una piccola ROM con i vettori di test mancanti
  - multiplexer condividere le uscite dell'LFSR e della ROM
  - i test set devono essere compattati
- Una tecnica alternativa si basa su un diffrattore che genera un cluster di pattern nelle vicinanze (Hamming) di ogni test contenuto nella ROM
- Reseeding: una piccola ROM contiene un certo numero di stati iniziali per l'LFSR in modo da esploare meglio lo spazio dei possibili vettori di ingresso al circuito da collaudare

Analisi della risposta

Analisi della risposta

- Tecniche costose: utilizzo di una ROM, duplicazione del circuito o compressione della risposta
- Utilizzo di compattatori
  - conteggio del numero di 1
  - conteggio del numero di transizioni
  - utilizzo di LFSR
- La risposta compattata o **signature** viene confrontata con quella del circuito corretto
- I compattatori perdono informazioni per cui alcune sequenze di uscita errate possono dare luogo alla stessa risposta compattata
- Questo problema viene definito **aliasing** e viene caratterizzato probabilisticamente

Compattazione della risposta

- Consideriamo per semplicitá un singolo PO
- Conteggio del numero di 1
  - data una sequenza con lunghezza L e k uni é indistinguibile da $\binom{L}{k}$ altre sequenze

$$\text{probabilitá di alias } = \frac{\binom{L}{k} - 1}{2^L - 1}$$

- Conteggio del numero di transizioni
  - data una sequenza con lunghezza L e k transizioni é indistinguibile da $2\binom{L}{k}$ altre sequenze

$$\text{probabilitá di alias} = \frac{2\binom{L}{k} - 1}{2^L - 1}$$

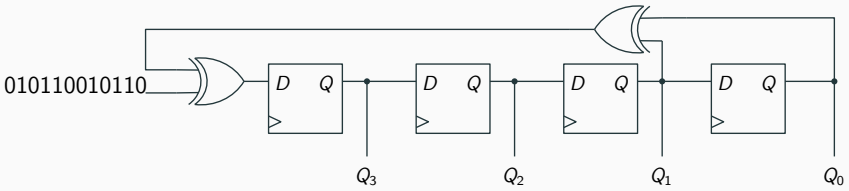
Ones count vs transition count

- In entrambi i casi bisogna massimizzare o minimizzare il conteggio in modo da ridurre il prodotto binomiale
- Chiaramente questo puó ridurre la copertura
- Nel caso del transition count é possibile riorganizzare le sequenze cosa che non é possibile nel conteggio di uni
  - ad esempio la sequenza di uscita 00010 10011 01011 00100 ha $L = 20$ e $k = 8$
  - nel conteggio degli uni si ha una probabilitá di alias pari 0.12
  - nel conteggio di transizioni si ha 0.24
  - se peró si riordina la sequenza di test come 00000 00000 00111 11111 si ha 0.000036
- Questa riorganizzazione peró rende chiaramente la sequenza efficacie solo rispetto a guasti statici e non a stuck-open o transition faults

Utilizzo di LFSR per la compattazione della risposta

- Tecnica sviluppata in HP per il collaudo sul campo di schede che viene denominata **signature analysis**
- I bit di uscita del circuito sono trattati come i coefficenti di un polinomio
- L'LFSR divide tale polinomio in ingresso per il suo polinomio caratteristico
- Il resto della divisione rimane nell'LFSR come signature
- L'LFSR va inizializzato a un valore noto prima del collaudo e una volta eseguito il collaudo il suo contenuto va confrontato con la signature corretta

Esempio di signature analysis



input polynomial	input	state $Q_3Q_2Q_1Q_0$	output polynomial	
	0	0000		← stato iniziale
x^{10}	1	0000		
	0	1000		
x^8	1	0100		
x^7	1	1010		
	0	0101	x^6	
	0	1010		
x^4	1	1101	x^4	
	0	0110		
x^2	1	1011	x^2	
x^1	1	1101	x^1	
	0	0110		← resto

si puó verificare che
$\mathbf{p = qd + r}$

polinomio caratteristico
$d = x^4 + x + 1$

polinomio di ingresso
$p = 0x^{11} + 1x^{10} + 0x^9 + 1x^8 + 1x^7 + 0x^6 + 0x^5 + 1x^4 + 0x^3 + 1x^2 + 1x + 0 = x^{10} + x^8 + x^7 + x^4 + x^2 + x$

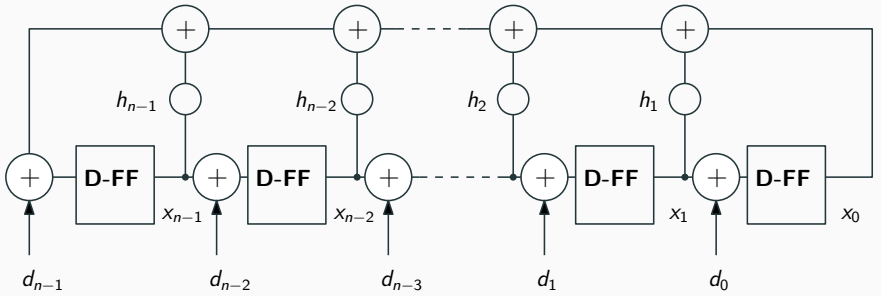
polinomio di uscita
$q = x^6 + x^4 + x^2 + x$

resto
$r = x^3 + x^2$

Multiple Input Shift Register (MISR)

- La probabilitá di alias diminuisce con l'aumentare del numero di FF utilizzati
- Utilizzare uno LFSR di dimensioni ragionevoli per ciascuna uscita implica un hardware overhead notevole
- Soluzione: MISR che compatta tutte le uscit con un unico LFSR
  - funziona perché l'LFSR é lineare e vale il principio di sovrapposizione
  - sovrappone tutte le risposte come se entrassero in un LFSR a ingresso singolo
  - il resto finale, che é la signature, é l'XOR dei resti delle divisioni polinomiale degli ingressi provenienti dai vari PO per i polinomio caratteristico

MISR



Equazione del MISR

$$\begin{bmatrix} x_0(t+1) \\ x_1(t+1) \\ . \\ . \\ . \\ x_{n-3}(t+1) \\ x_{n-2}(t+1) \\ x_{n-1}(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & .... & 0 & 0 \\ 0 & 0 & 1 & .... & 0 & 0 \\ . & . & . & & . & . \\ . & . & . & & . & . \\ . & . & . & & . & . \\ 0 & 0 & 0 & .... & 1 & 0 \\ 0 & 0 & 0 & .... & 0 & 1 \\ 1 & h_1 & h_2 & .... & h_{n-2} & h_{n-1} \end{bmatrix} \begin{bmatrix} x_0(t) \\ x_1(t) \\ . \\ . \\ . \\ x_{n-3}(t) \\ x_{n-2}(t) \\ x_{n-1}(t) \end{bmatrix} + \begin{bmatrix} d_0(t) \\ d_1(t) \\ . \\ . \\ . \\ d_{n-3}(t) \\ d_{n-2}(t) \\ d_{n-1}(t) \end{bmatrix}$$

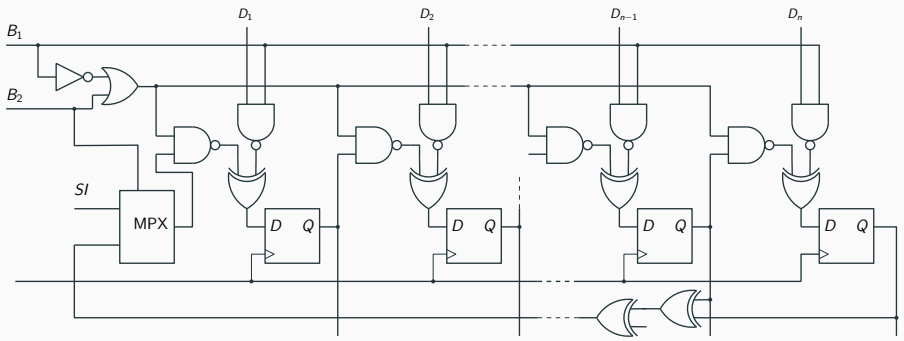
Built-In Logic Block Observer BILBO

- Componente per pseudorandom testing in grado di
  - funzionare come un normale registro
  - funzionare come uno shift-register per inizializzare il componente ed estrarre la signature
  - funzionare come un LFSR per generare sequenze pseudo casuali
  - funzionare da MISR per raccogliere la signature

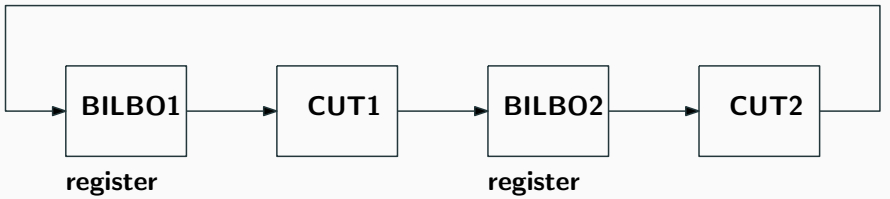
BILBO

Codifica delle operazioni:

B_1B_2	operazione
00	SR
01	LFSR
10	R
11	MISR

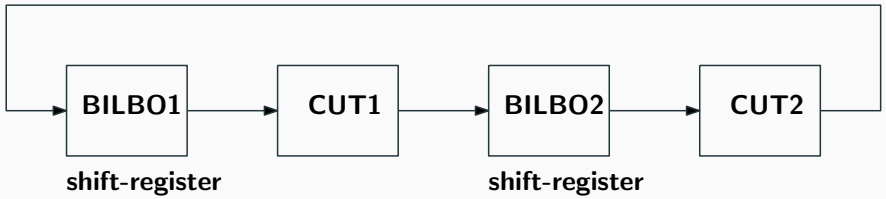


Pseudorandom testing con BILBO



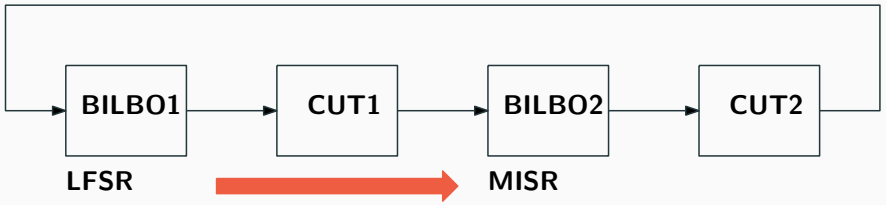
Normal mode

Pseudorandom testing con BILBO



Initialize BILBO1 and BILBO2

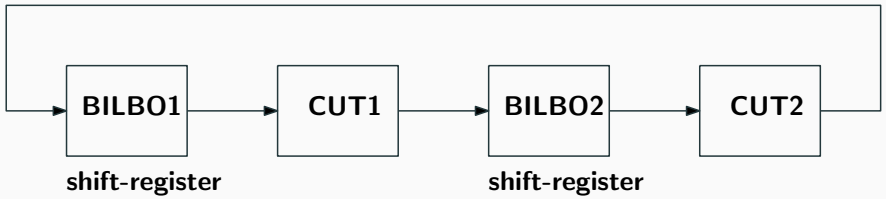
Pseudorandom testing con BILBO



Test CUT1

Introduzione Hardware pattern generator Tecniche per migliorare un TPG hardware Analisi della risposta 36

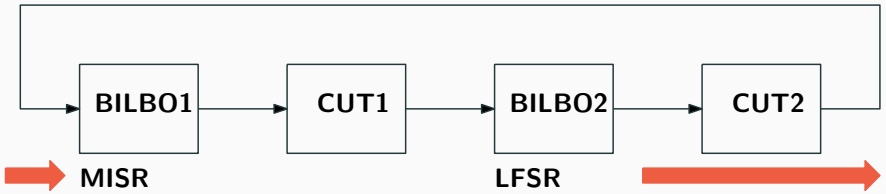
Pseudorandom testing con BILBO



Initialize BILBO1, extract signature from BILBO2 while initializing it

Introduzione Hardware pattern generator Tecniche per migliorare un TPG hardware Analisi della risposta 36

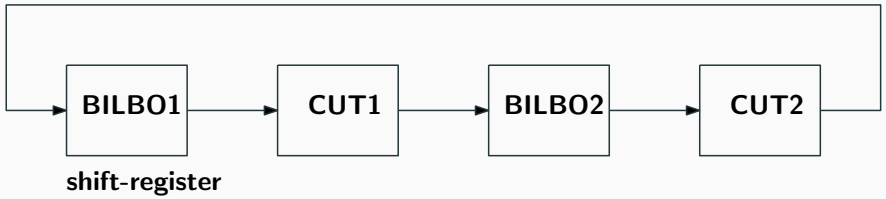
Pseudorandom testing con BILBO



Test CUT2

Introduzione Hardware pattern generator Tecniche per migliorare un TPG hardware Analisi della risposta 36

Pseudorandom testing con BILBO



Extract signature from BILBO1