Chapter 5

Logic Built-In Self-Test

What is this chapter about?

- Introduce the basic concepts of logic BIST
- BIST Design Rules
- Test pattern generation and output response analysis techniques
- Fault Coverage Enhancement
- Various BIST timing control diagrams
- A Design Practice

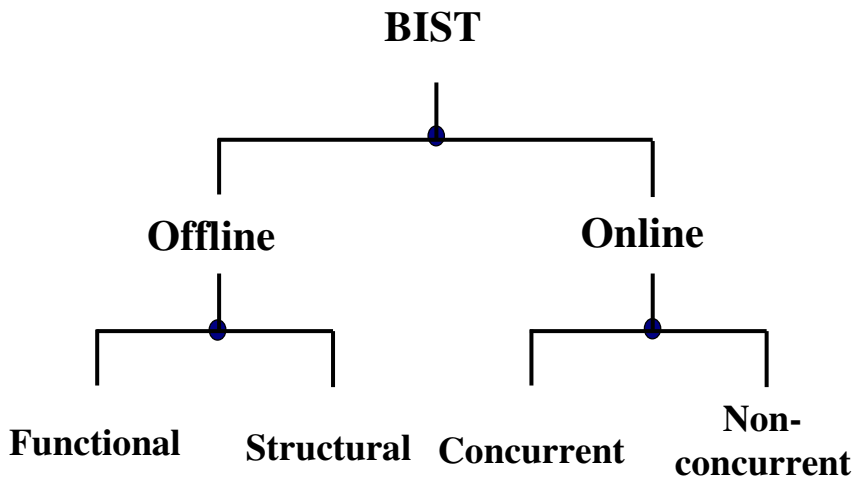
Introduction

- What are the problems in today's semiconductor testing?
  - Traditional test techniques become quite expensive
  - No longer provide sufficiently high fault coverage
- Why do we need built-in self-test (BIST)?
  - For mission-critical applications
  - Detect un-modeled faults
  - Provide remote diagnosis

BIST Techniques Categories

- Online BIST
  - Concurrent online BIST
  - Non Concurrent online BIST
- Offline BIST
  - Functional offline BIST
  - Structural offline BIST

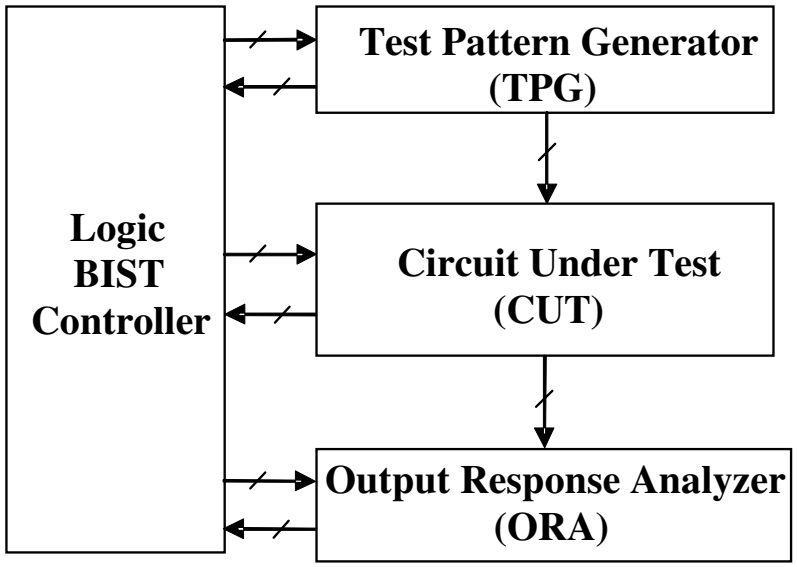
A General Form of Logic BIST



[Abramovici 1994]

Logic BIST Techniques

A Typical Logic BIST System

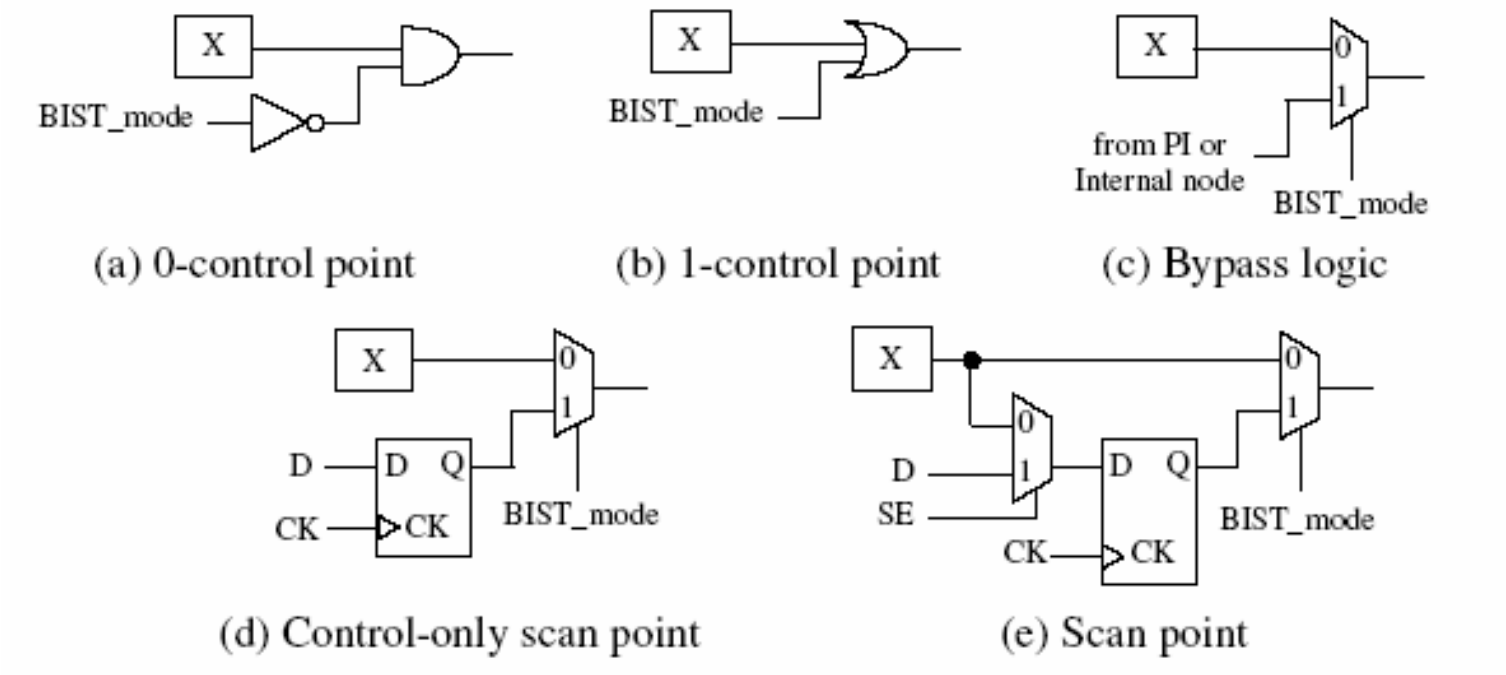


Structural off-line BIST

BIST Design Rules

Logic BIST requires much more stringent design restrictions when compared to conventional scan. Therefore, when designing a logic BIST system, it is essential that the circuit under test meet all *scan design rules* and *BIST specific design rules*, called BIST design rules.

Typical X-bounding Methods



Methods for blocking an unknown (X) source

X-bounding Methods

Depending on the nature of each unknown (X) source, several X-bounding methods can be appropriate for use.

Common problems:

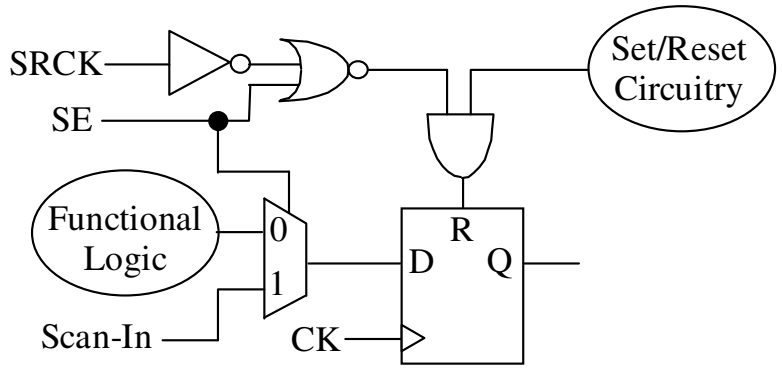
(1) Increase the area of the design.
(2) Impact timing.

Typical Unknown Sources

- **Analog Blocks**
  - Adding bypass logic.
  - Adding control-only scan point
- **Memories and Non-Scan Storage Elements**
  - Bypass logic
  - Initialization
- **Combinational Feedback Loops**
  - Scan points

Typical Unknown Sources (cont'd)

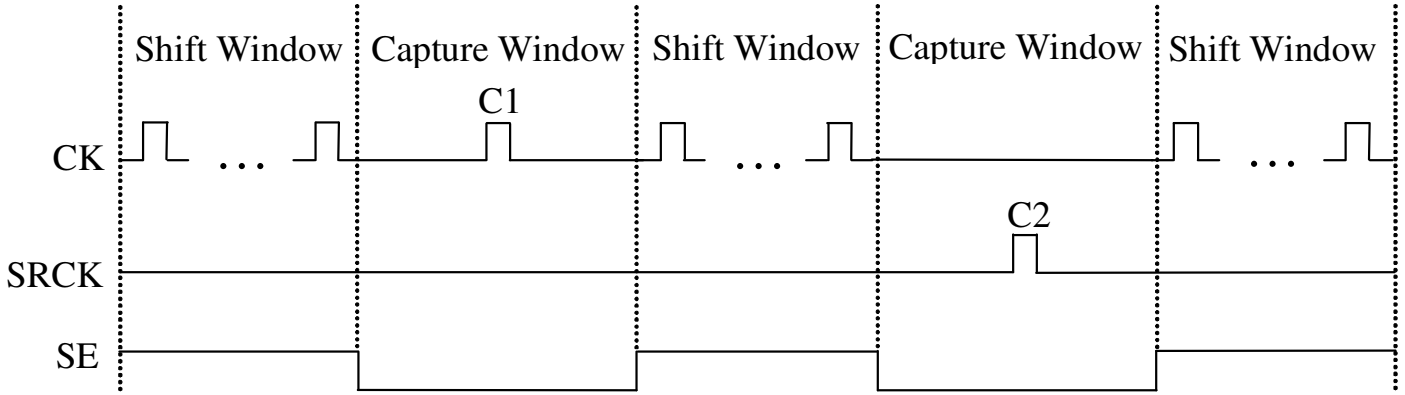
- **Asynchronous Set/Reset Signals**
  - using the existing scan enable (SE) signal to protect each shift operation and adding a **set/reset clock point** (SRCK) on each set/reset signal to test the set/reset circuitry.



[Abdel-Hafez 2004]

Typical Unknown Sources (cont'd)

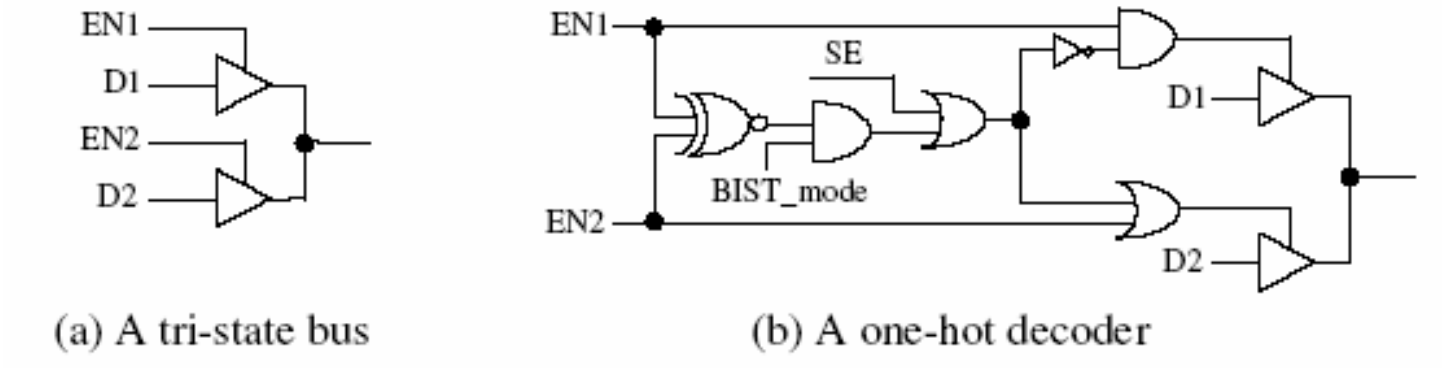
- Asynchronous Set/Reset Signals



Timing control diagram for testing data and set/reset faults

Typical Unknown Sources (cont'd)

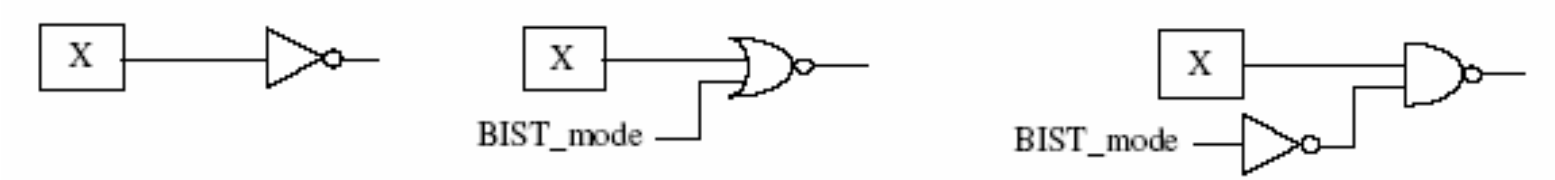
- **Tri-State Buses**
  - Re-synthesize each bus with multiplexers.
  - One-hot decoder



A one-hot decoder for testing a tri-state bus with 2 drivers

Typical Unknown Sources (cont'd)

- **False Paths**
  - 0-control point
  - 1-control point
- **Critical Paths**
  - Adding an extra input pin to a selected combinational gate on the critical path.



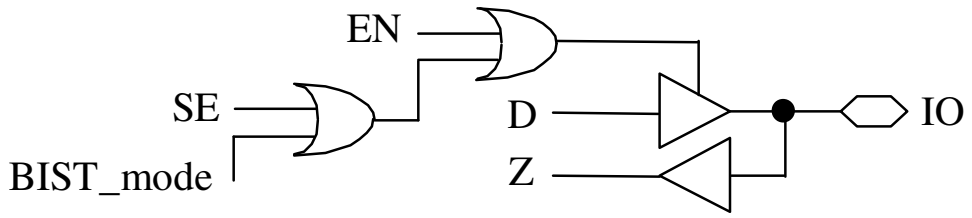
(a) An inverter (b) Embedded 0-control point (c) Embedded 1-control point

Typical Unknown Sources (cont'd)

- **Multiple-Cycle Paths**
  - 0-control point
  - 1-control point
  - Holding certain scan cell output states
- **Floating Ports**
  - PI or PO must have a proper connection to Power (Vcc) or Ground (Vss).
  - Floating inputs to any internal modules must be avoided.

Typical Unknown Sources (cont'd)

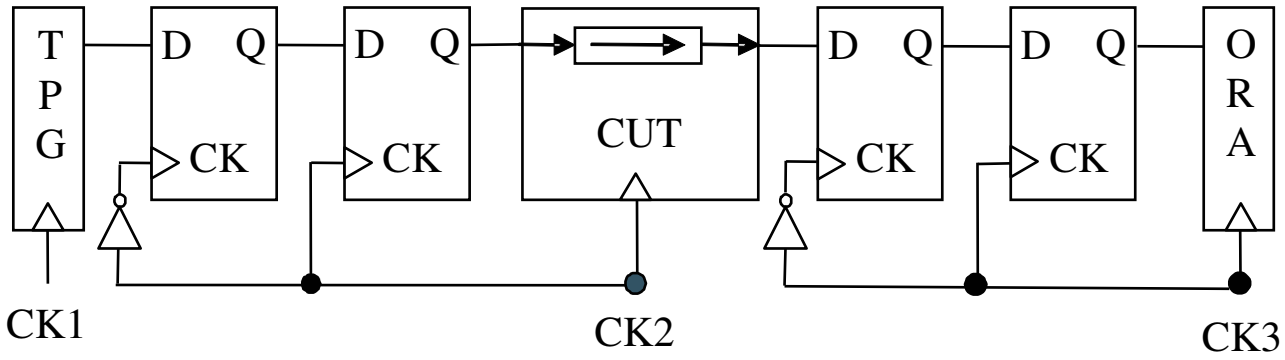
- **Bi-directional I/O Ports**
  - Fix the direction of each bi-directional I/O port to either input or output mode.



Forcing a bi-directional port to output mode

Re-Timing

Races and hazards caused by clock skews may occur between the TPG and the (scan chain) inputs of the CUT as well as between the (scan chain outputs of the CUT and the ORA. To avoid these potential problems and ease physical implementation, we recommend adding *re-timing* logic between the TPG and the CUT and between the CUT and the ORA.



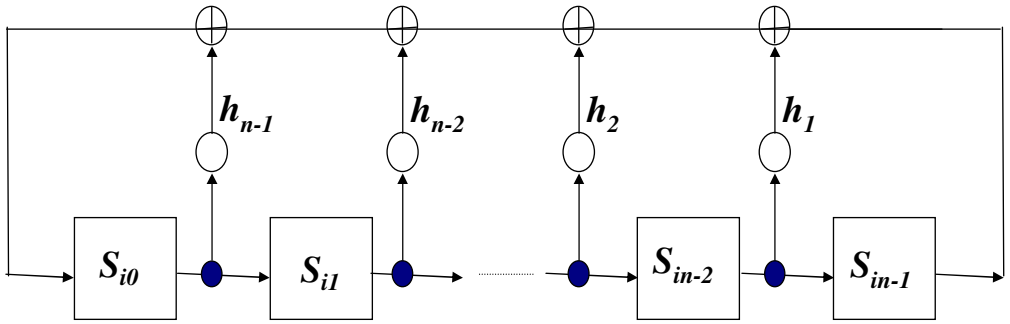
Re-timing logic among the TPG, CUT, and ORA

Test Pattern Generation

- Test pattern generators (TPGs) constructed from linear feedback shift registers (LFSRs)
- TPG
  - Exhaustive testing
  - Pseudo-random testing
  - Pseudo-exhaustive testing

Standard LFSR

- Consists of *n* D flip-flops and a selected number of exclusive-OR (XOR) gates

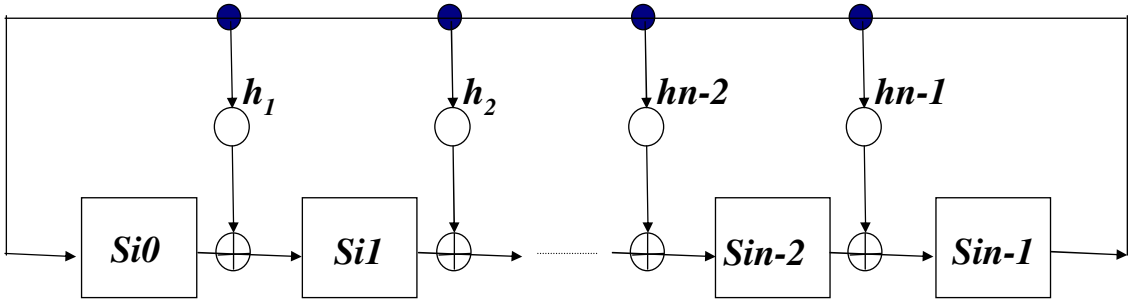


[Golomb 1982]

An n-stage (external-XOR) standard LFSR

Modular LFSR

- Each XOR gate placed between two adjacent D flip-flops



[Golomb 1982]

An n-stage (internal-XOR) modular LFSR

LFSR Properties

- The internal structure of the *n*-stage LFSR can be described by a characteristic polynomial of degree *n*, *f(x)*.

$$f(x) = 1 + h_1x + h_2x^2 + \ldots + h_{n-1}x^{n-1} + x^n.$$

h_i is either 1 or 0,depending on the feedback path

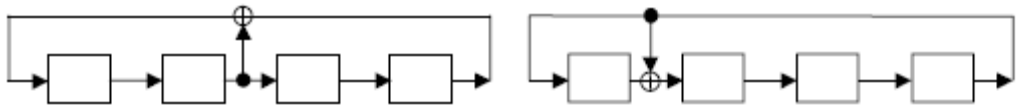
LFSR Properties

- Let S_i represent the contents of the n-stage LFSR after i^{th} shifts of the initial contents, S_0, of the LFSR, and $S_i(x)$ be the polynomial representation of S_i

$$S_i(x) = S_{i0} + S_{i1}x + S_{i2}x^2 + \ldots + S_{in-2}x^{n-2} + S_{in-1}x^{n-1}.$$

If T is the smallest positive integer such that f(x) divides $1 + x^T$ *, then the integer T is called the period of the LFSR.*

4-stage standard and modular LFSRs



(a) A 4-stage standard LFSR

(b) A 4-stage modular LFSR

(c) Test sequence generated by (a)	(d) Test sequence generated by (b)
0 0 0 1	**0 0 0 1**
1 0 0 0	1 1 0 0
0 1 0 0	0 1 1 0
1 0 1 0	0 0 1 1
0 1 0 1	1 1 0 1
0 0 1 0	1 0 1 0
0 0 0 1	0 1 0 1
1 0 0 0	1 1 1 0
0 1 0 0	0 1 1 1
1 0 1 0	1 1 1 1
0 1 0 1	1 0 1 1
0 0 1 0	1 0 0 1
0 0 0 1	1 0 0 0
1 0 0 0	0 1 0 0
0 1 0 0	0 0 1 0
1 0 1 0	**0 0 0 1**

- 4-stage Standard LFSR
- 4-stage Modular LFSR

$$f(x) = 1 + x + x^4$$

$$s_0 = x^3$$

Hybrid LFSR

$$a(x) = 1 + b(x) + c(x)$$

Fully decomposable iff both b(x) and c(x) have no common terms and there exists an integer j such that $c(x) = x^j b(x),\ j \geq 1$

Assume: f(x) is fully decomposable

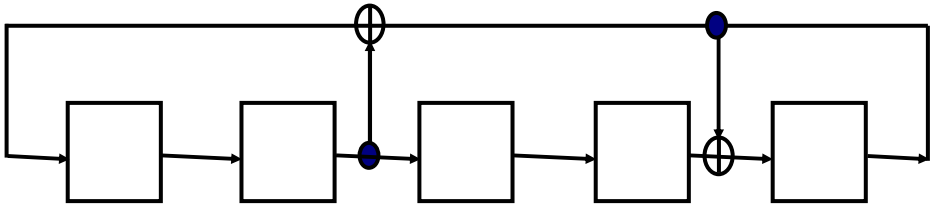
$$f(x) = 1 + b(x) + x^j b(x)$$

A (hybrid) top-bottom LFSR [Wang 1988a] can be constructed:

$$s(x) = 1 + \hat{}\, x^j + x^j b(x)$$

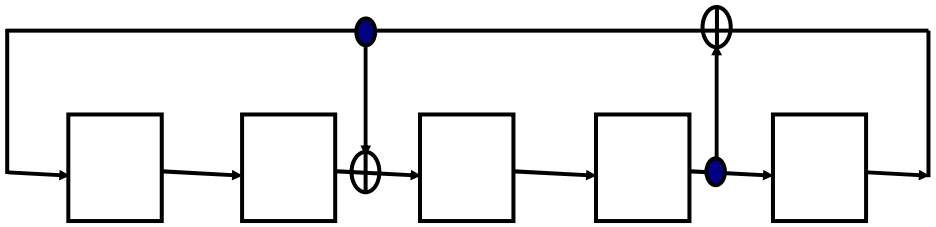
Indicate the XOR gate with one input Is connected to the feedback path, not between stages

5-stage hybrid LFSRs



$s(x) = 1+^x^2+x^4+x^5$ for $f(x) = 1+x^2+x^3+x^4+x^5$

(a) 5-stage top-bottom LFSR



$s(x) = 1+x^2+^x^4+x^5$ for $f(x) = 1+x+x^2+x^3+x^5$

(b) 5-stage bottom-top LFSR

Primitive polynomials list

Primitive polynomials of degree n up to 100

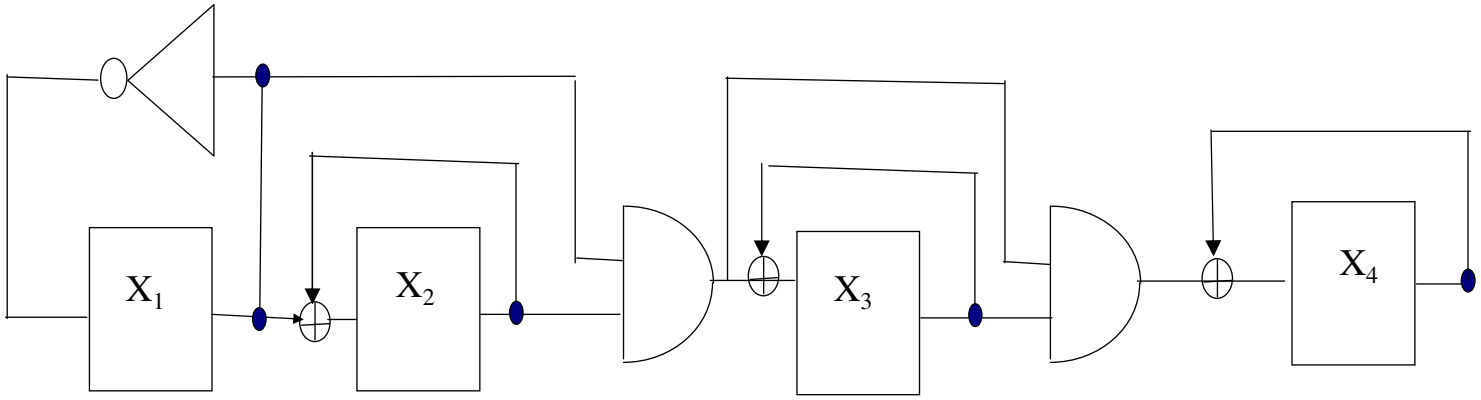
n	*Exponents*	*n*	*Exponents*	*n*	*Exponents*	*n*	*Exponents*
1	0	26	8 7 1 0	51	16 15 1 0	76	36 35 1 0
2	1 0	27	8 7 1 0	52	3 0	77	31 30 1 0
3	1 0	28	3 0	53	16 15 1 0	78	20 19 1 0
4	1 0	29	2 0	54	37 36 1 0	79	9 0
5	2 0	30	16 15 1 0	55	24 0	80	38 37 1 0
6	1 0	31	3 0	56	22 21 1 0	81	4 0
7	1 0	32	28 27 1 0	57	7 0	82	38 35 3 0
8	6 5 1 0	33	13 0	58	19 0	83	46 45 1 0
9	4 0	34	15 14 1 0	59	22 21 1 0	84	13 0
10	3 0	35	2 0	60	1 0	85	28 27 1 0
11	2 0	36	11 0	61	16 15 1 0	86	13 12 1 0
12	7 4 3 0	37	12 10 2 0	62	57 56 1 0	87	13 0
13	4 3 1 0	38	6 5 1 0	63	1 0	88	72 71 1 0
14	12 11 1 0	39	4 0	64	4 3 1 0	89	38 0
15	1 0	40	21 19 2 0	65	18 0	90	19 18 1 0
16	5 3 2 0	41	3 0	66	10 9 1 0	91	84 83 1 0
17	3 0	42	23 22 1 0	67	10 9 1 0	92	13 12 1 0
18	7 0	43	6 5 1 0	68	9 0	93	2 0
19	6 5 1 0	44	27 26 1 0	69	29 27 2 0	94	21 0
20	3 0	45	4 3 1 0	70	16 15 1 0	95	11 0
21	2 0	46	21 20 1 0	71	6 0	96	49 47 2 0
22	1 0	47	5 0	72	53 47 6 0	97	6 0
23	5 0	48	28 27 1 0	73	25 0	98	11 0
24	4 3 1 0	49	9 0	74	16 15 1 0	99	47 45 2 0
25	3 0	50	27 26 1 0	75	11 10 1 0	100	37 0

Note: "24 4 3 1 0" means $p(x) = x^{24} + x^4 + x^3 + x^1 + x^0$

Exhaustive Testing

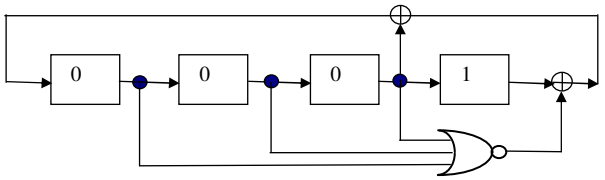
- Exhaustive Testing
  - Applying 2^n exhaustive patterns to an *n*-input combinational *circuit under test* (CUT)
- Exhaustive pattern generator
  - Binary counter
  - Complete LFSR

Binary counter

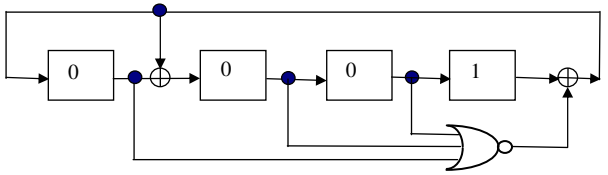


Example binary counter as EPG

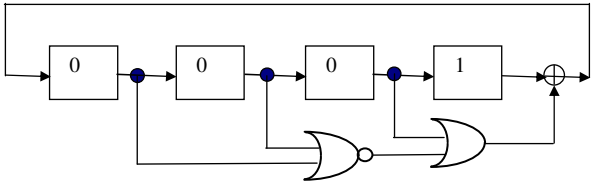
Complete LFSR



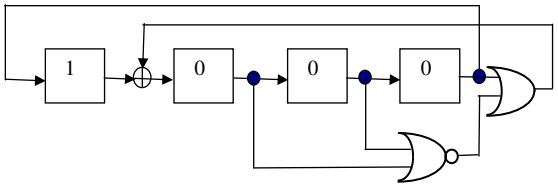
(a) 4-stage standard CFSR



(b) 4-stage modular CFSR



(c) A minimized version of (a)



(d) A minimized version of (b)

Example complete LFSRs as EPG

Exhaustive Testing performance

- Exhaustive Testing guarantees all detectable, combinational faults will be detected.
- Test time maybe be prohibitively long if input number is large than 20.

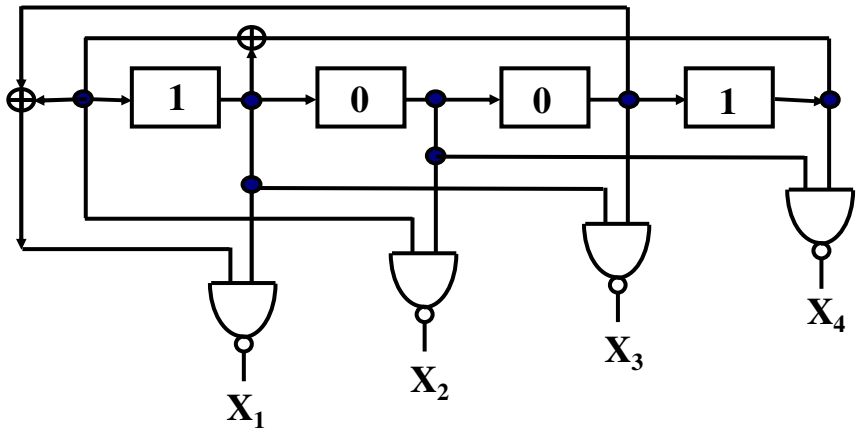
Pseudo-Random Testing

- Pseudo-random pattern generator
- Reduce test length but sacrifice the fault coverage
- Difficult to determine the required test length and fault coverage

Pseudo-Random Testing

- Maximum-length LFSR
  - RP-resistant problem
- Weighted LFSR
- Cellular Automata

Weighted LFSR



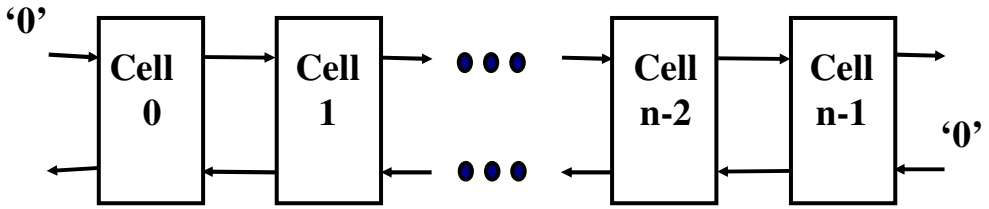
Example weighted LFSR as PRPG

Cellular Automata

- Provide more random test patterns
- Provide high fault coverage in a random-pattern resistant (RP-resistant) circuit
- Implementation advantage

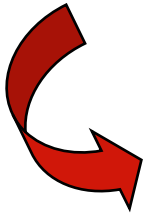
Cellular Automata

A general structure of an *n*-stage CA



Rule 90: $x_i(t+1) = x_{i-1}(t) + x_{i+1}(t)$,

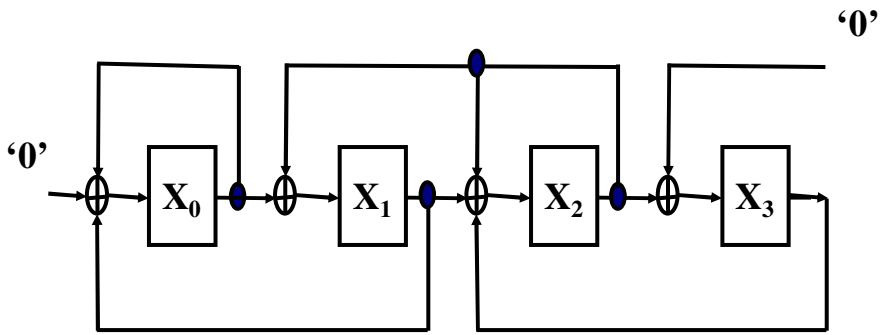
Rule 150: $x_i(t+1) = x_{i-1}(t) + x_i(t) + x_{i+1}(t)$.



Each rule determines the next state of a cell based on the state of the cell and its neighbors

Example cellular automaton

A 4-stage CA



Test sequence

0 0 0 1
0 0 1 0
0 1 1 1
1 1 1 1
0 0 1 1
0 1 0 1
1 0 0 0
1 1 0 0
0 1 1 0
1 1 0 1
0 1 0 0
1 0 1 0
1 0 1 1
1 0 0 1
1 1 1 0

CA construction rules

Construction rules for cellular automata of length *n* up to 53

n	*Rule **	*n*	*Rule **	*n*	*Rule **	*n*	*Rule **
4	05	17	175,763	30	7,211,545,075	43	035,342,704,132,622
5	31	18	252,525	31	04,625,575,630	44	074,756,556,045,302
6	25	19	0,646,611	32	10,602,335,725	45	151,315,510,461,515
7	152	20	3,635,577	33	03,047,162,605	46	0,112,312,150,547,326
8	325	21	3,630,173	34	036,055,030,672	47	0,713,747,124,427,015
9	625	22	05,252,525	35	127,573,165,123	48	0,606,762,247,217,017
10	0,525	23	32,716,432	36	514,443,726,043	49	02,675,443,137,056,631
11	3,252	24	77,226,526	37	0,226,365,530,263	50	23,233,006,150,544,226
12	2,525	25	136,524,744	38	0,345,366,317,023	51	04,135,241,323,505,027
13	14,524	26	132,642,730	39	6,427,667,463,554	52	031,067,567,742,172,706
14	17,576	27	037,014,415	40	00,731,257,441,345	53	207,121,011,145,676,625
15	44,241	28	0,525,252,525	41	15,376,413,143,607		
16	152,525	29	2,512,712,103	42	11,766,345,114,746		

[Hortensius 1989]

**For n=7, Rule=152=001,101,010=1,101,010, where "0" denotes a rules 90 and "1" denotes a rule 150 cell, or vice versa*

Pseudo-Exhaustive Testing

- Reduce test time while retaining many advantages of exhaustive testing
- Guarantee 100% single-stuck fault coverage
  - Verification test technique *[McCluskey 1984]*
  - Segmentation test technique *[McCluskey 1981]*

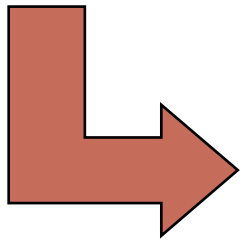
Verification Testing

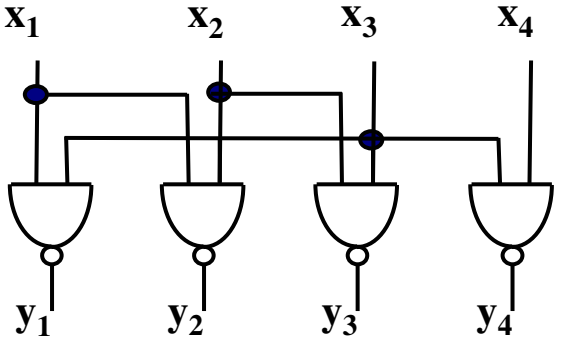
Divide the CUT into m cones, backtracing from each output to determine the inputs that drive the output. Each cone will receive exhaustive test patterns and are tested concurrently.

[McCluskey 1984]

Pseudo-exhaustive pattern generators

PEPGs



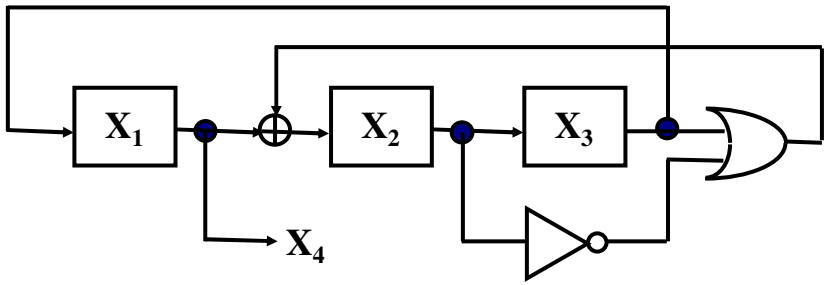


An (n, w)=(4, 2) CUT

Syndrome Driver Counter

Use SDC to generate test patterns. Check whether some inputs can share the same test signal. If *n-p* Inputs can share test inputs with other *p* inputs, then the circuit can be tested exhaustively with these *p* inputs.

[Savir 1980]

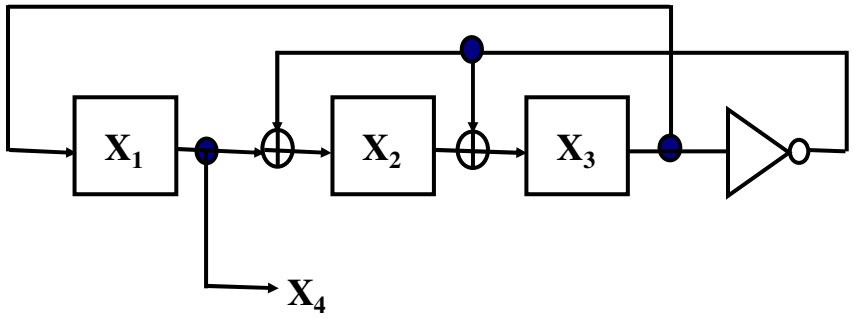


A 3-stage syndrome driver counter

Constant-Weight Counter

Use CWCs to generate test patterns. Constant-Weight counters are constructed using *constant-weight code* or *M-out-of-N code.* The constant-weight test set is a minimum-length test set for many circuits.

[McCluskey 1982]

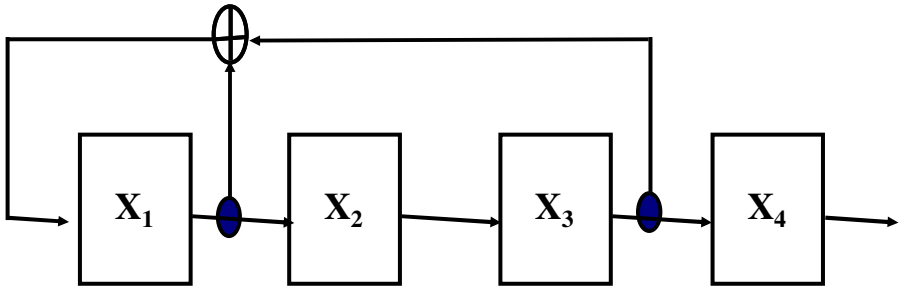


A 3-stage constant-weight counter

Combined LFSR/SR

Use a combination of an LFSR and a shift register (SR) for pattern generation. The method is most effective when *w* is much less than *n*. In general, this technique requires much more tests than other schemes when *w* is greater than *n/2*.

[Barzilai 1983] [Tang 1984]

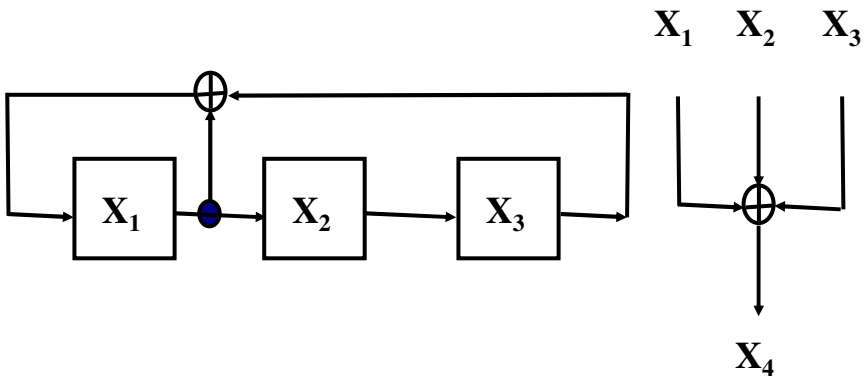


A 4-stage combined LFSR/SR

Combined LFSR/PS

A combined LFSR/PS approach using a combination of an LFSR and a linear phase shifter which includes a network of XOR gates to generate test pattern. Similar to combined LFSR/SR, this technique requires more tests than other schemes when *w* is greater than *n/2*.

[Vasanthavada 1985]



A 3-stage combined LFSR/PS

Condensed LFSR

Condensed LFSRs are constructed based on linear codes. Define *g(x)* and *p(x)* as the generator polynomial and primitive polynomial over GF(2), respectively. An *(n, k)* condensed LFSR can be realized using

$$f(x) = g(x)\,p(x) = (1 + x + x^2 + \ldots + x^{n-k})\,p(x)$$

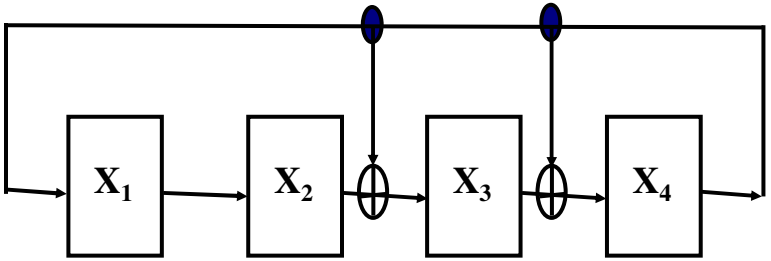
Where

$$w < [k/(n-k+1)] + [k/(n-k+1)]$$

[Wang 1986a]

Example Condensed LFSR

A (4,3) condensed LFSR



Test sequence

1100
0110
0011
1010
0101
1001
1111

Set

$$S_0(x) = g(x) = 1+x, \text{ or } S_0 \text{ to } \{1100\}$$

Cyclic LFSR

Use cyclic LFSRs to reduce the test length when *w < n/2.*

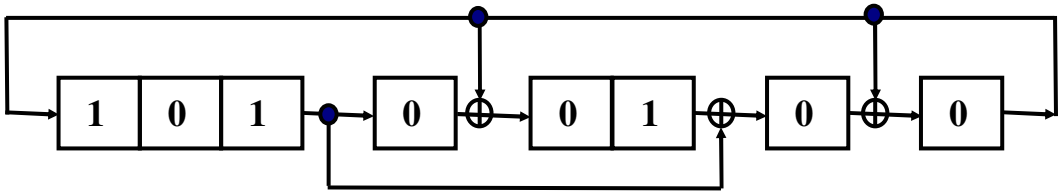
A cyclic code always exists when $n' = 2^b - 1, b > 1$

To exhaustively test any *(n,w)* CUT

- find a generator polynomial *g(x)* of largest degree k' (or smallest degree k), for generating an *(n',k') = (n',n'-k)* cyclic code, that divides 1+x^^n' and has a design distance *d > w+1*;
- construct an (n',k) cyclic LFSR using f(x) = h(x)p(x) = (1+x^^n')p(x)/g(x), where h(x) = (1+x^^n')/g(x); and
- shorten this *(n',k)* cyclic LFSR to an *(n,k)* cyclic LFSR by deleting the rightmost, middle, or leftmost *n'-n* stages from the *(n',k)* cyclic LFSR.

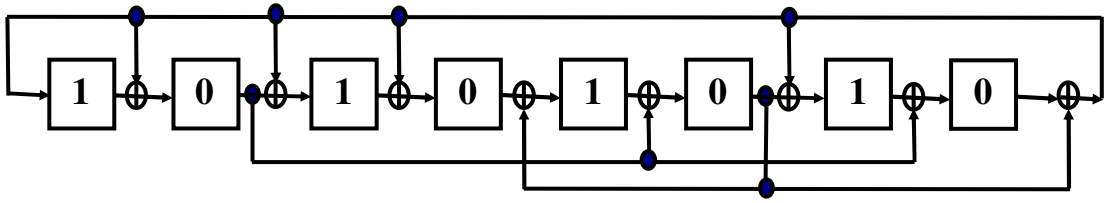
Example Cyclic LFSR

A (8,5) cyclic LFSR, picking the first 6 stages and the last two stages of the (15,5) cyclic LFSR.



[Wang 1988b]

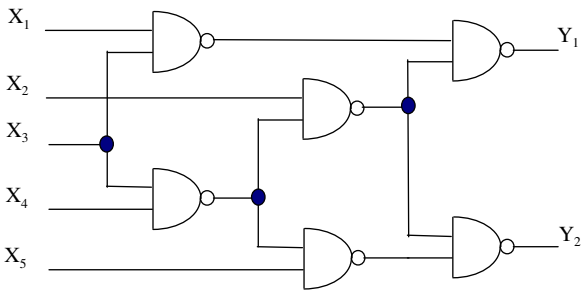
An (n,k-s) shorted cyclic LFSR can be employed when $n = 2^b, b > 2$



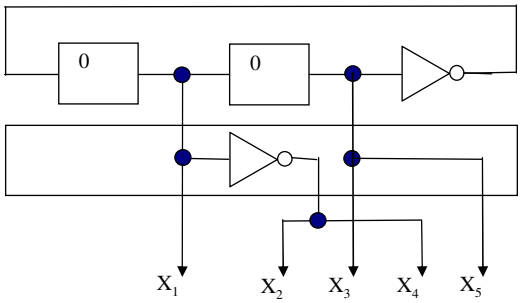
[Wang 1987b]

Compatible LFSR

The combined LFSR of an *l*-stage LFSR and an *l-to-n* mapping logic, called *l*-stage compatible LFSR, can further reduce the test length, when only single stuck faults are considered.



(a) An (n,w) = (5,4) CUT



(b) A 2-stage compatible LFSR

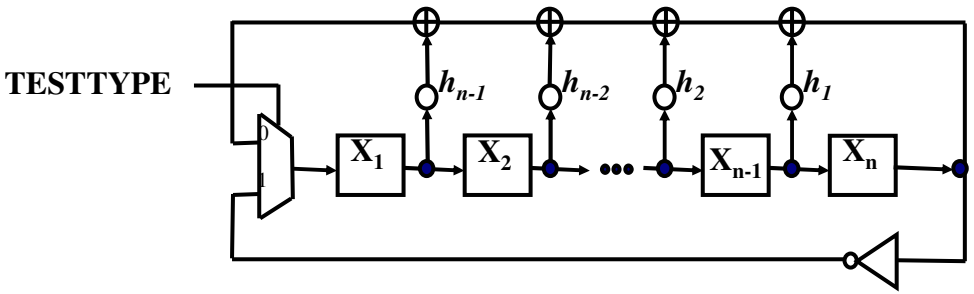
Example compatible LFSR as PEPG

Segmentation Testing

- Used when
  - Test length using previous techniques is too long or
  - Output depends on all inputs.
- Divide the circuit into segments
  - Hardware partitioning
  - Sensitized partitioning

Delay Fault Testing

- Need $2^n(2^n - 1)$ patterns to test delay fault exhaustively
- Test set could cause test invalidation when more than one inputs change.



[Bushnell 2000]

Output Response Analysis

- Ones count testing
- Transition count testing
- Signature analysis

Ones Count Testing

Assume the CUT has one output and the output contains a stream of *L* bits. Let the fault-free output response be

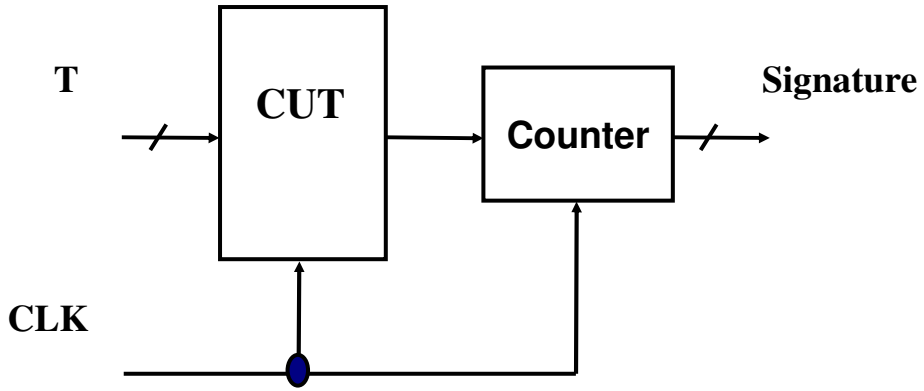
$$\{r_0, r_1, r_2 \cdots r_{L-1}\}$$

Ones count testing will need a counter to count the number of 1s in the bit stream.

Aliasing probability *[Savir 1985]*

$$P_{OC}(m) = (C(L,m) - 1)/(2^L - 1)$$

One Count Testing



One counter as ORA

Transition Count Testing

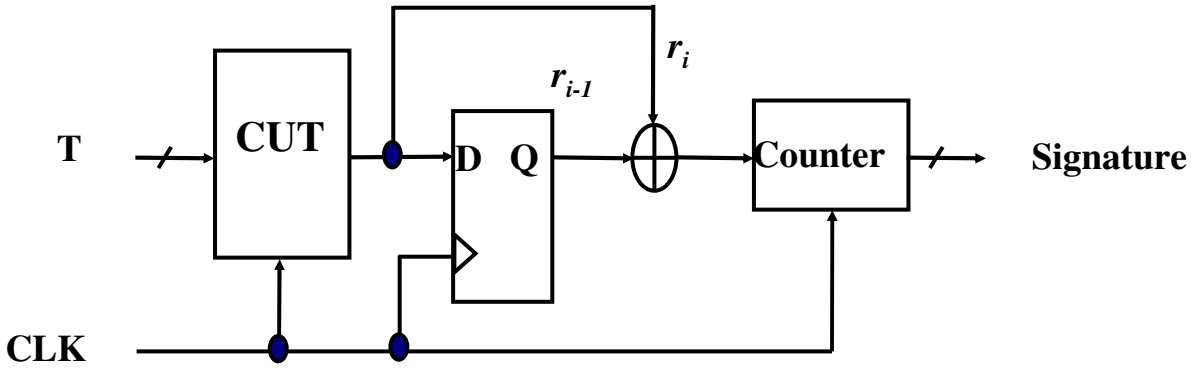
Transition count testing is similar to that for ones count testing, except the signature is defined as the number of *1-to-0* and *0-to-1 transitions.*

[Hayes 1976]

Aliasing probability

$$P_{TC}(m) = (2C(L-1,m)-1)/(2^{L}-1)$$

Transition Count Testing



Transition counter as ORA

Signature Analysis

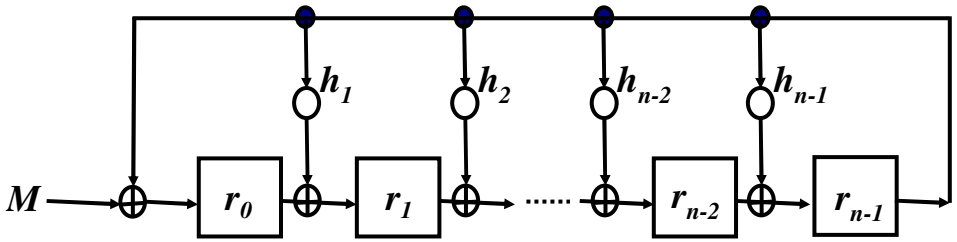
Signature analysis is the most popular compaction technique used today, based on cyclic redundancy checking.

Two signature analysis schemes

- Serial signature analysis
- Parallel signature analysis

Serial Signature Analysis

An n-stage single-input signature register



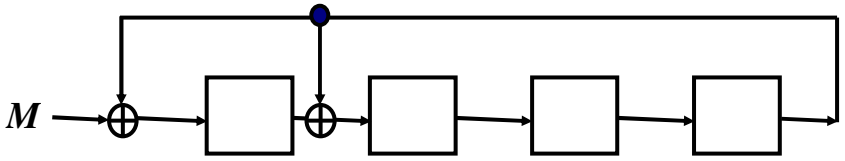
Define *L-bit* output sequence *M*

$$M(x) = m_0 + m_1 x + m_2 x + \ldots + m_{L-1} x^{L-1}$$

Let the polynomial of the modular be *f(x)*

IF $M(x) = q(x) f(x) + r(x)$ ⟹ Signature is the polynomial remainder, *r(x)*

Example



M	r_0	r_1	r_2	r_3
1	**0**	**0**	**0**	**0**
1	1	0	0	0
0	1	1	0	0
1	0	1	1	0
1	1	0	1	1
0	0	0	0	1
0	1	1	0	0
1	0	1	1	0
R	**1**	**0**	**1**	**1**

(a) Fault-free signature

M'	r_0	r_1	r_2	r_3
1	**0**	**0**	**0**	**0**
1	1	0	0	0
0	1	1	0	0
1	0	1	1	0
0	1	0	1	1
0	1	0	0	1
1	1	0	0	0
1	1	1	0	0
R'	**1**	**1**	**1**	**0**

(b) Signature for fault f_1

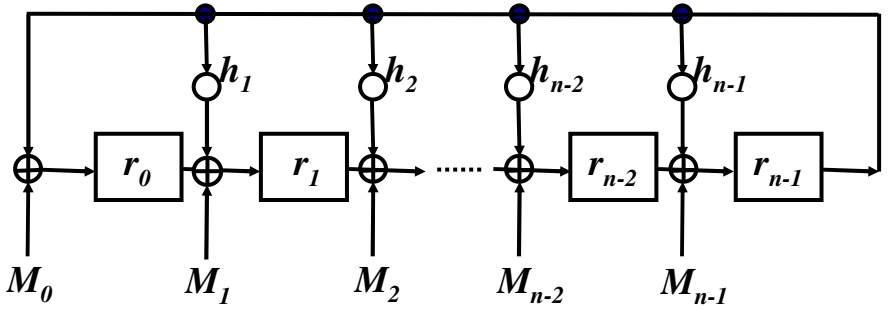
M''	r_0	r_1	r_2	r_3
1	**0**	**0**	**0**	**0**
0	1	0	0	0
1	0	1	0	0
1	1	0	1	0
0	1	1	0	1
0	1	0	1	0
1	0	1	0	1
1	0	1	1	0
R''	**1**	**0**	**1**	**1**

(c) Signature for fault f_2

A 4-stage SISR

Parallel Signature Analysis

Multiple-input signature register (MISR)

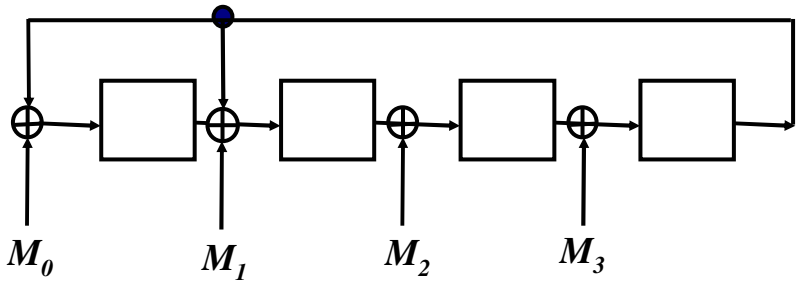


An n-input MISR can be remodeled as a single-input SISR with *effective input sequence M(x)* and *effective error polynomial E(x)*

$$M(x) = M_0(x) + xM_1(x) + \ldots + x^{n-2}M_{n-2}(x) + x^{n-1}M_{n-1}(x)$$

$$E(x) = E_0(x) + xE_1(x) + \ldots + x^{n-2}E_{n-2}(x) + x^{n-1}E_{n-1}(x)$$

4-stage MISR



A 4-stage MISR

M_0	1 0 0 1 0
M_1	0 1 0 1 0
M_2	1 1 0 0 0
M_3	1 0 0 1 1
M	1 0 0 1 1 0 1 1

An equivalent M sequence

Aliasing probability

$$P_{PSA}(n) = (2^{(mL-n)} - 1)/(2^{mL} - 1)$$

Logic BIST Architectures

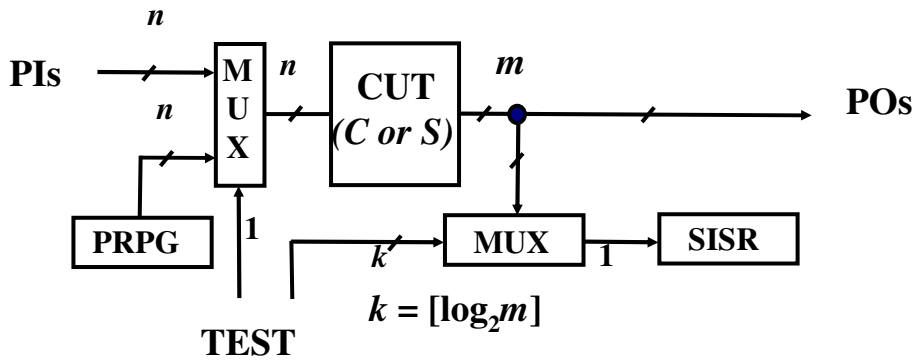
Four Types of BIST Architectures:

- No special structure to the CUT
- Make use of scan chains in the CUT
- Configure the scan chains for test pattern generation and output response analysis
- Use concurrent checking circuitry of the design

Type I - *Centralized and Separate Board-Level BIST (CSBL)*

Two LFSRs and two multiplexers are added to the circuit.
The first LFSR acts as a PRPG, the second serves as a SISR.
The first multiplexer selects the inputs, another routes the PO to the SISR.

[Benowitz 1975]

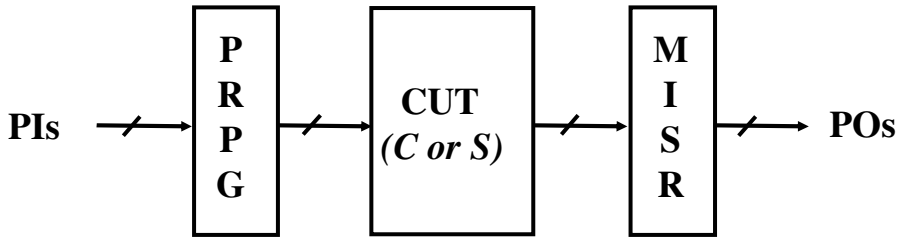


CSBL Architecture

Type I - *Built-In Evaluation and Self-Test (BEST)*

Use a PRPG and a MISR. Pseudo-random patterns are applied in parallel from the PRPG to the chip primary inputs (PIs) and a MISR is used to compact the chip output responses .

[Perkins 1980]

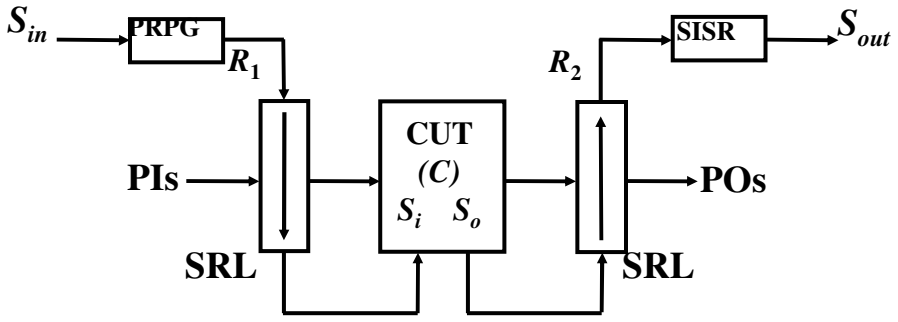


BEST Architecture

Type II - *LSSD On-Chip Self-Test (LOCST)*

In addition to the internal scan chain, an external scan chain comprising all primary inputs and primary outputs is required. The External scan-chain input is connected to the scan-out point of the internal scan chain.

[Eichelberger 1983]



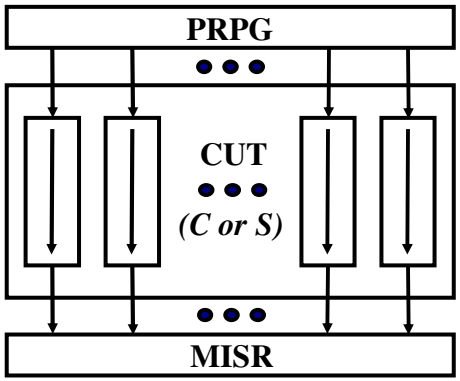
LOCST Architecture

Type II - *Self-Testing Using MISR and Parallel SRSG (STUMPS)*

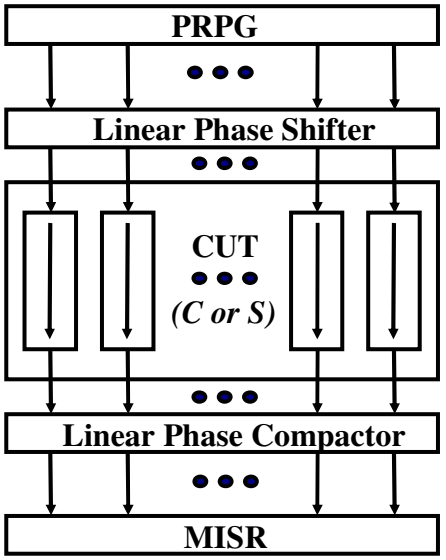
Contains a PRPG (SRSG) and a MISR. The scan chains are loaded in parallel from the PRPG. The system clocks are then pulsed and the test responses are scanned out to the MISR for compaction. New test patterns are scanned in at the same time when the test responses are being scanned out.

[Bardell 1982]

STUMPS



STUMPS



A STUMPS-based Architecture

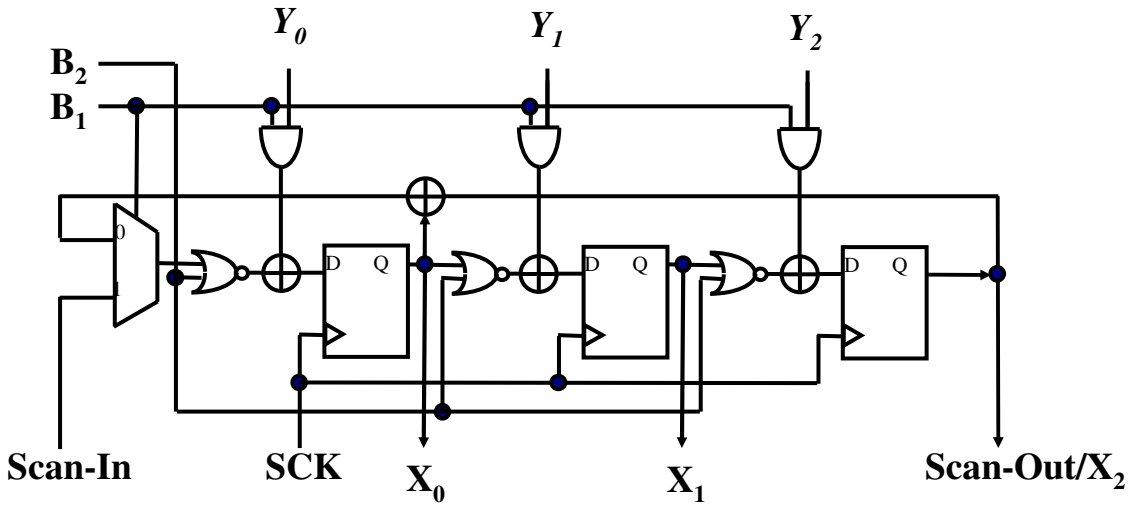
Type III - *Built-In Logic Block Observer (BILBO)*

The architecture applies to circuits that can be partitioned into independent modules (logic blocks). Each module is assumed to have its own input and output registers (storage elements), or such registers are added to the circuit where necessary. The registers are redesigned so that for test purposes they act as PRPGs or MISRs.

[Konemann 1980]

Built-In Logic Block Observer

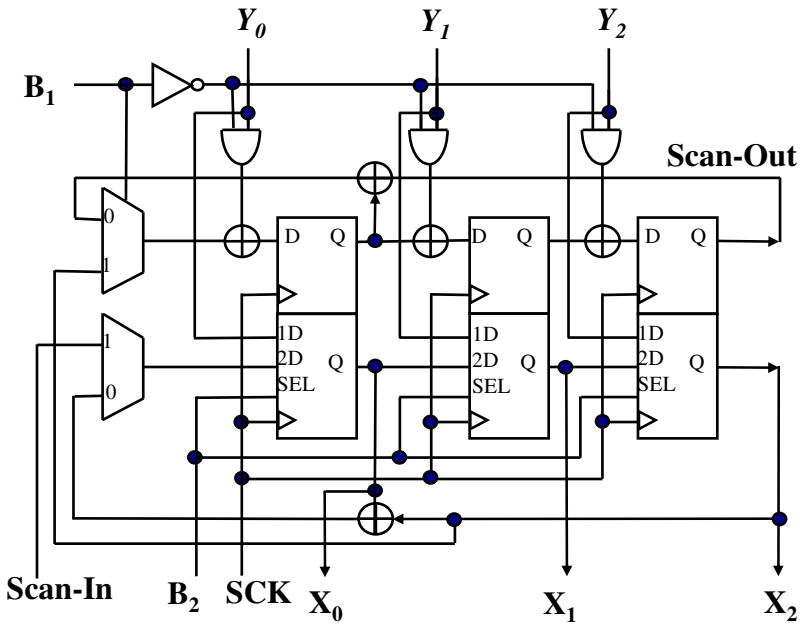
B_1	B_2	Operation mode
1	1	Normal
0	0	Scan
1	0	Mixed Test Generation and Signature Analysis
0	1	Reset



A 3-stage BILBO

Type III - *Concurrent Built-In Logic Block Observer (CBILBO)*

B_1	B_2	Operation mode
-	0	Normal
1	1	Scan
0	1	Test Generation and Signature Analysis



[Wang 1986c]

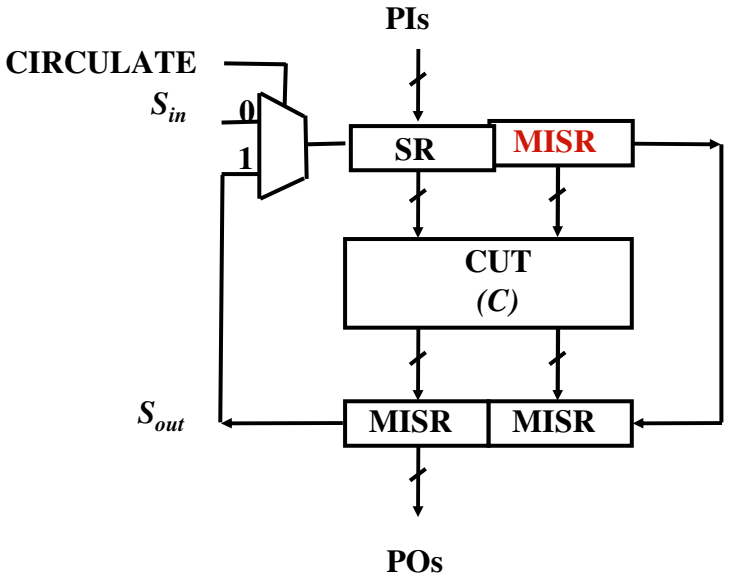
A 3-stage concurrent BILBO (CBILBO)

Type III - *Circular Self-Test Path (CSTP)*

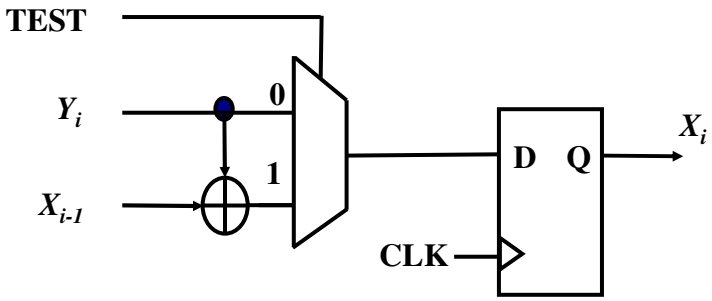
All primary inputs and primary outputs are reconfigured as external scan cells. They are connected to the internal scan cells to form a circular path. During self-test, all primary inputs (PIs) are connected as a shift register (SR), whereas all internal scan cells and primary outputs (POs) are reconfigured as a MISR.

[Krasniewsk 1989]

Circular Self-Test Path



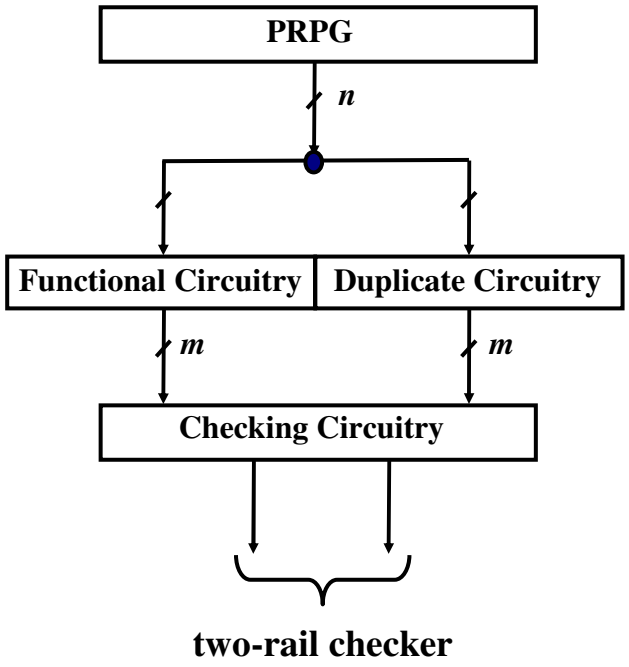
(a) The CSTP architecture



(b) Self-Test cell

CSTP architecture

Type IV - *Concurrent Self-Verification (CSV)*



CSV Architecture

Summary

Architecture	*Level*	*TPG*	*ORA*	*Circuit*	*BIST*
CSBL	B or C	PRPG	SISR	C or S	Test-Per-Clock
BEST	B or C	PRPG	MISR	C or S	Test-Per-Clock
LOCST	C	PRPG	SISR	C	Test-Per-Scan
STUMPS	B or C	PRPG	MISR	C	Test-Per-Scan
BILBO	C	PRPG	MISR	C	Test-Per-Clock
CBILBO	C	EPG/PEPG	MISR	C	Test-Per-Clock
CSTP	C	PRPG	MISR	C or S	Test-Per-Clock
CSV	C	PRPG	Checker	C or S	Test-Per-Clock

B: board-level testing
C: combinational circuit
S: sequential circuit

Representative Logic BIST Architectures

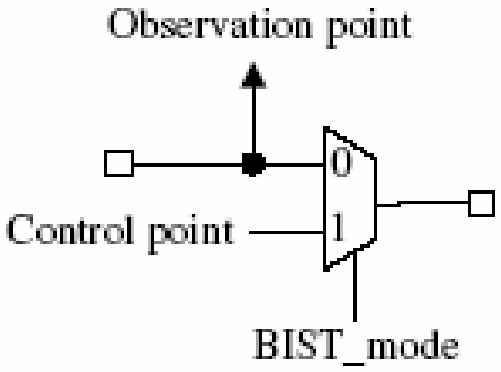
Fault Coverage Enhancement

Three approaches to enhance the fault coverage

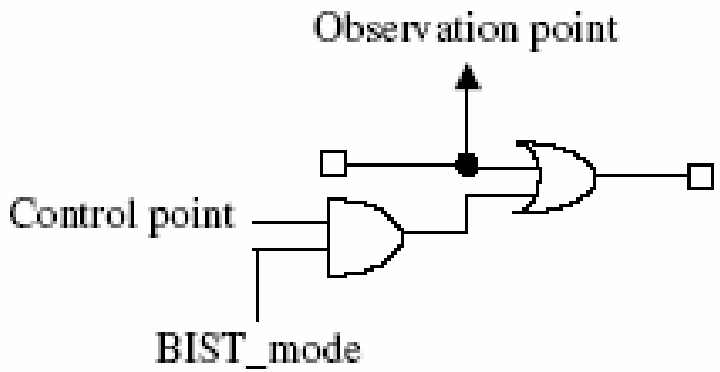
- Test point insertion
- Mixed-mode BIST
- Hybrid BIST

Test Point Insertion

Two typical types of test points



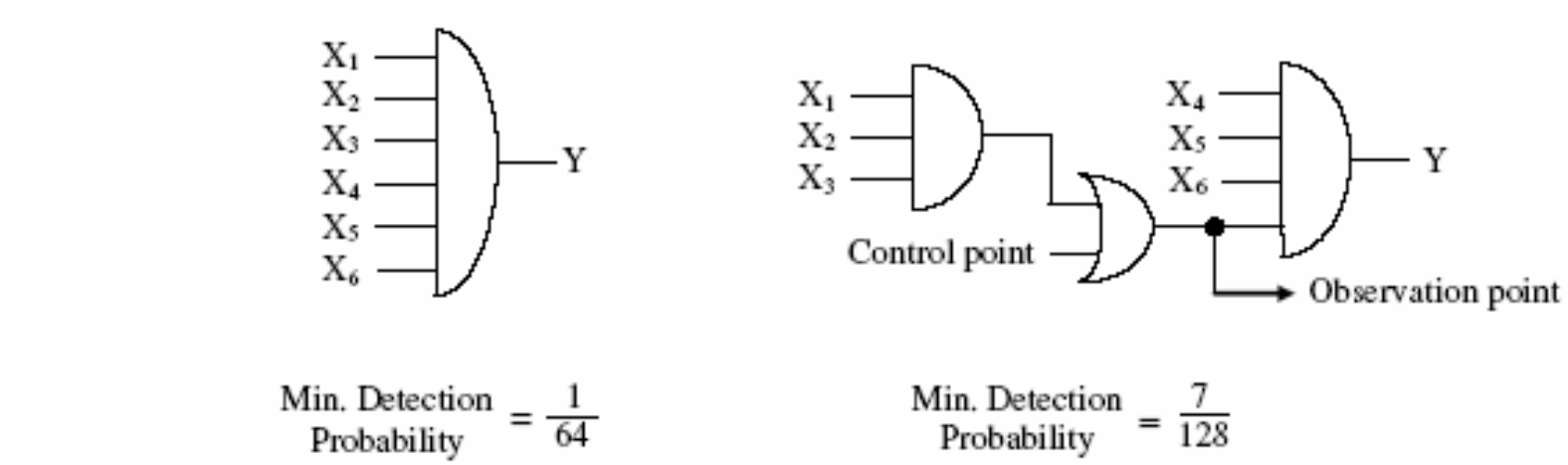
(a) Test point with a multiplexer



(b) Test point with AND-OR gates

Test Point Insertion Example

An example where one control point and one observation point are inserted to increase the detection probability of a 6-input AND-gate.



(a) An output RP-resistant stuck-at-0 fault (b) Example inserted test points

Test Point Placement

Where to place the test points in the circuit to maximize the coverage and minimize the number of test points required.

- Fault simulation guided techniques
- Testability measure guided techniques
- *Timing-driven test point insertion* techniques

Control Point Activation

During normal operation, all control points must be deactivated. During testing, there are different strategies as to when and how the control points are activated.

- Random activation
- Deterministic activation

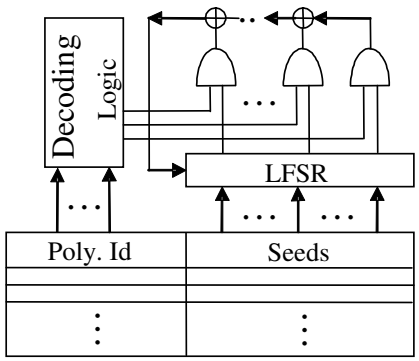
Mixed-Mode BIST

Mixed-mode BIST is an alternative way to improve fault coverage without modifying the CUT.
Pseudo-random patterns are generated to detect the RP-testable faults, and then some additional deterministic patterns are generated to detect the RP-resistant faults.

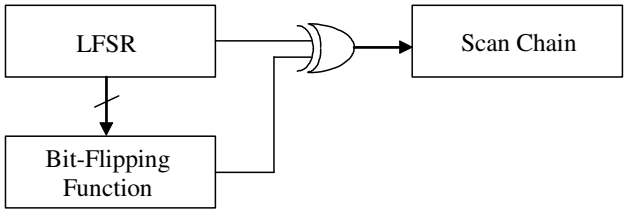
Mixed-Mode BIST

Approaches for generating deterministic patterns on-chip:

- ROM Compression.
- LFSR Reseeding.
- Embedding Deterministic Patterns.



Reseeding with multiple-polynomial LFSR



Bit-flipping BIST

Hybrid BIST

For manufacturing fault coverage enhancement where a tester is present, deterministic data from the tester can be used to improve the fault coverage.

- Top-up ATPG
- Store the compressed deterministic patterns on the tester

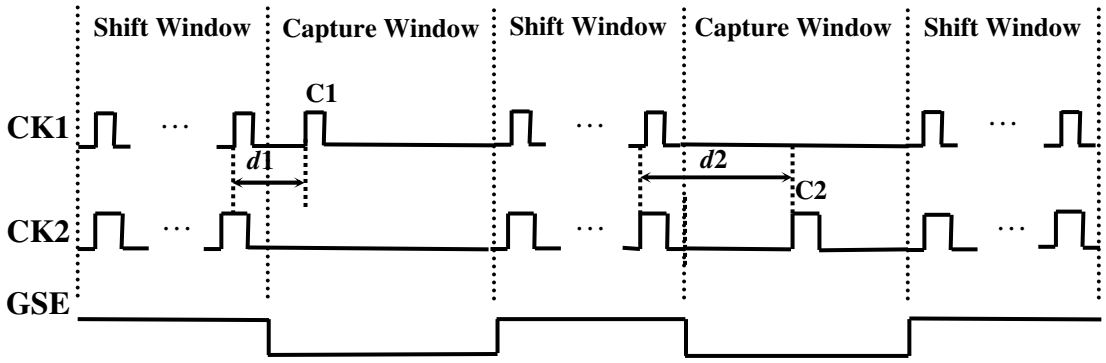
BIST Timing Control

- To test Multiple-clock-domain circuits
- To detect Intra-clock-domain faults and inter-clock-domain faults
- Capture-clocking schemes
  - Single-capture
  - Skewed-load
  - Double-capture

One-Hot Single-Capture

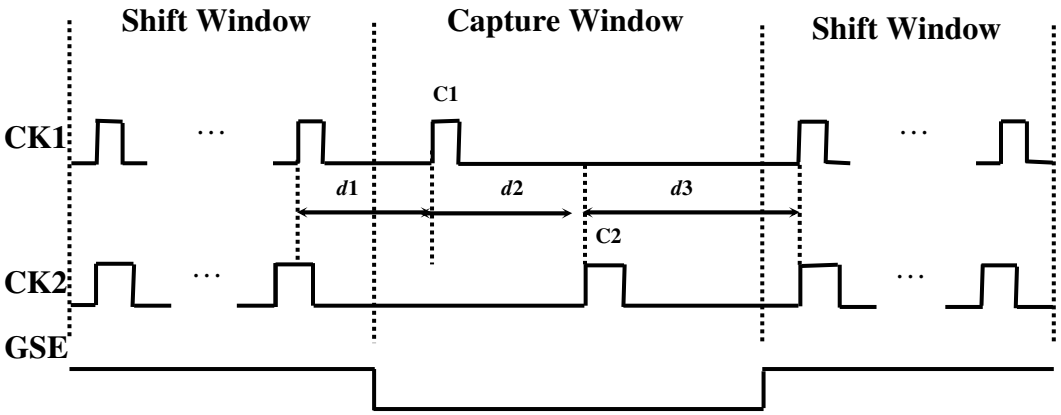
A capture pulse is applied to one clock domain, while holding all other test clocks inactive, during each capture window.

Benefit: a single and slow global scan mode signal
Drawback: long test time



One-hot single-capture

Staggered Single-Capture



Staggered single-capture

Benefits: short test time; a single and slow global scan mode signal
Drawback: some structural fault coverage loss

Skewed-Load

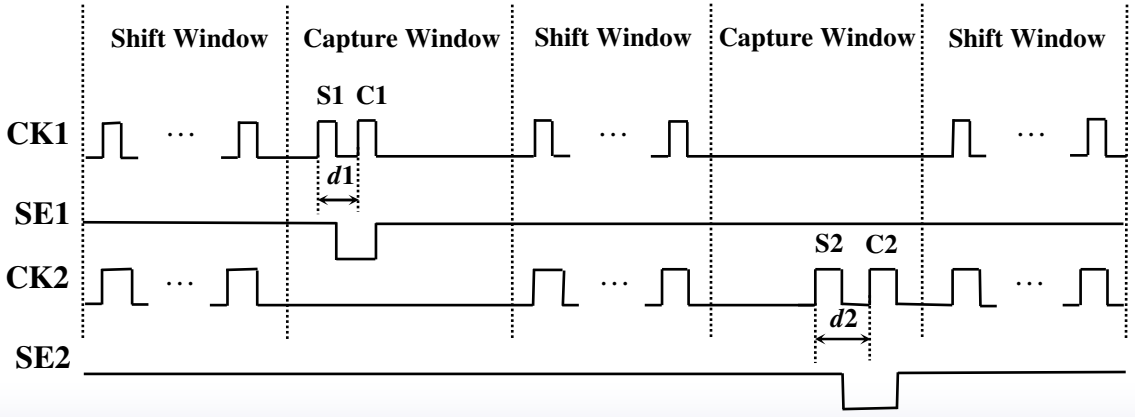
- An at-speed delay test technique
- Address intra-clock-domain delay faults
- Three approaches
  - One-hot skewed-load
  - Aligned skewed-load
  - Staggered skewed-load

One-Hot Skewed-Load

Tests all clock domains one by one by applying a-shift-followed by-a-capture pulses to detect intra-clock-domain delay faults.

Drawbacks:

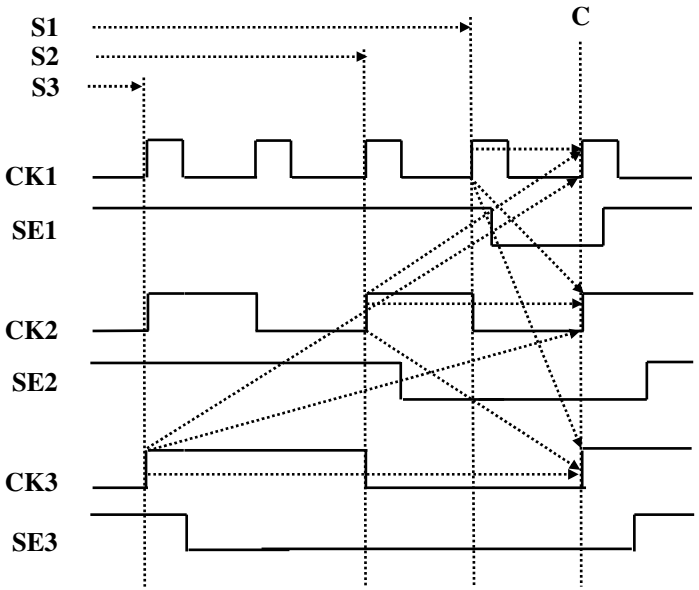
(1) Cannot detect inter-clock-domain delay faults
(2) Test time is long
(3) Single and global scan enable (GSE) signal can no longer be used



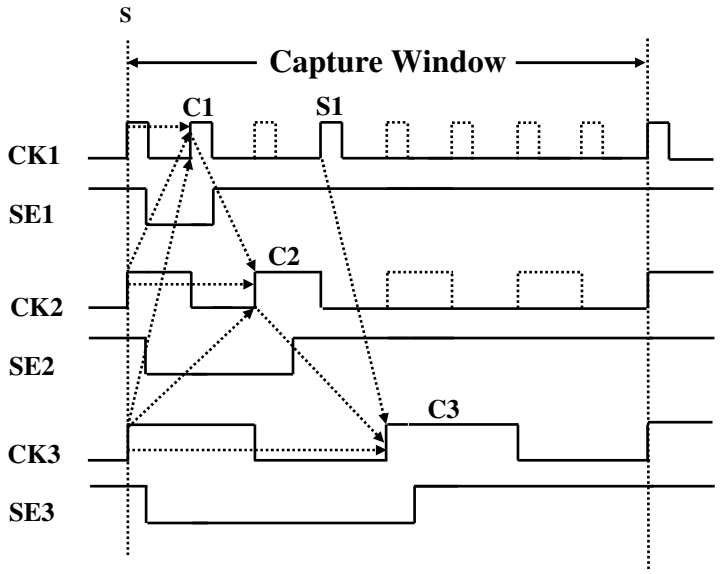
Aligned Skewed-Load

- Solve the long test time problem
- Test all intra-clock-domain and inter-clock-domain faults
- Need complex timing-control

Aligned Skewed-Load



Capture aligned skewed-load

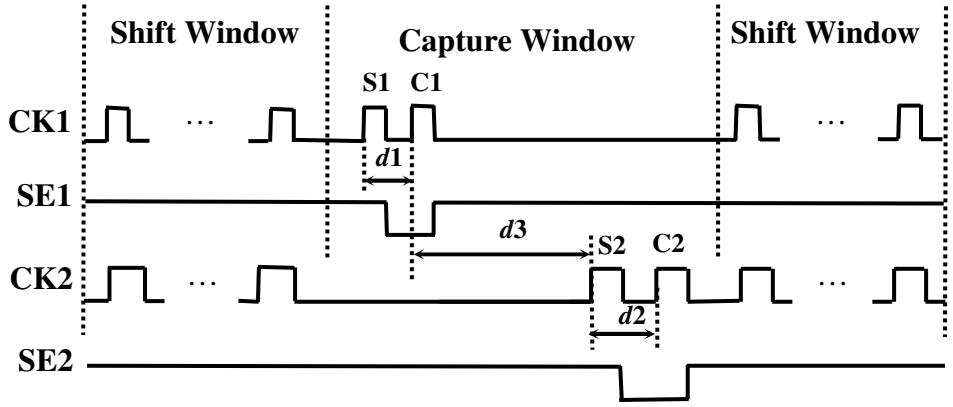


Launch aligned skewed-load

Staggered Skewed-Load

When two test clocks cannot be aligned precisely, we can simply insert a proper delay to eliminate the clock skew. The two last shift pulses are used to create transitions and their output responses are caught by the next two capture.

Drawback: Need at-speed scan enable signal for each clock domain



Staggered skewed-load

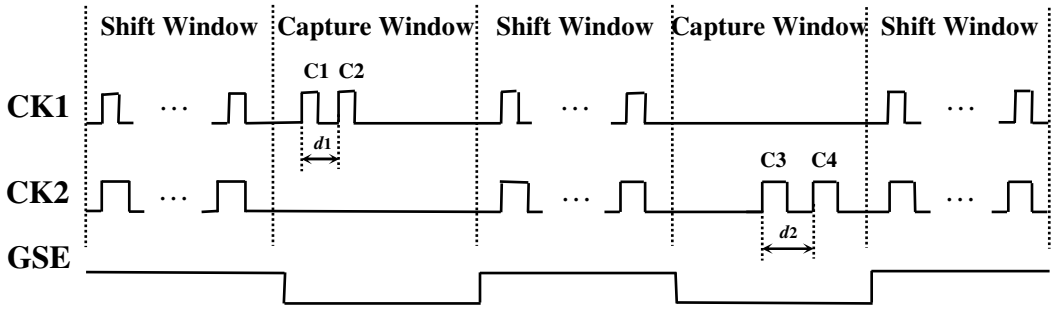
Double Capture

- Solve the physical implementation difficulty using skewed-load
- True at-speed test
- Double-capture benefits
  - Detect intra-clock-domain faults and inter-clock-domain structural faults or delay faults at-speed
  - Facilitate physical implementation
  - Ease integration with ATPG

One-Hot Double-Capture

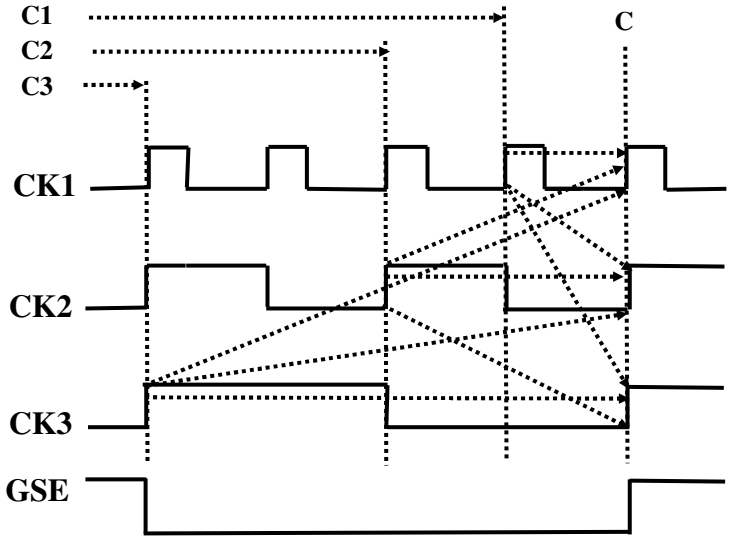
Test all clock domains one by one by applying two consecutive capture pulses at their respective domains' frequencies to test intra-clock-domain delay faults.

Benefit: true at-speed testing of intra-clock-domain delay faults
Drawbacks: (1) Cannot detect inter-clock-domain delay faults
(2) Test time is long

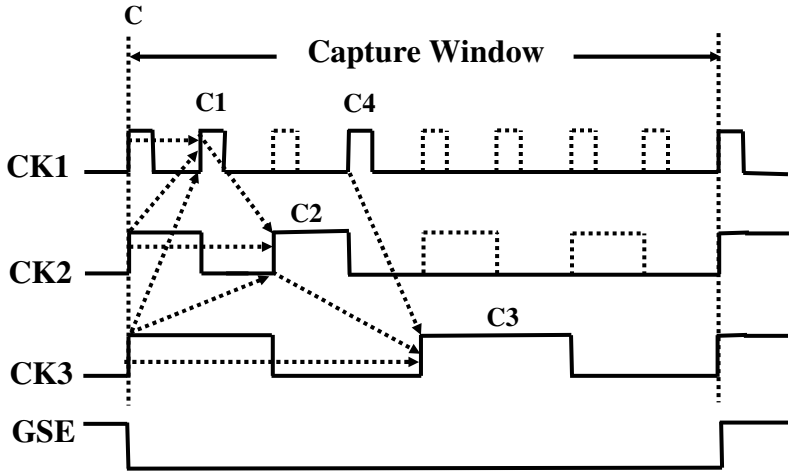


One-Hot double-capture

Aligned Double-Capture



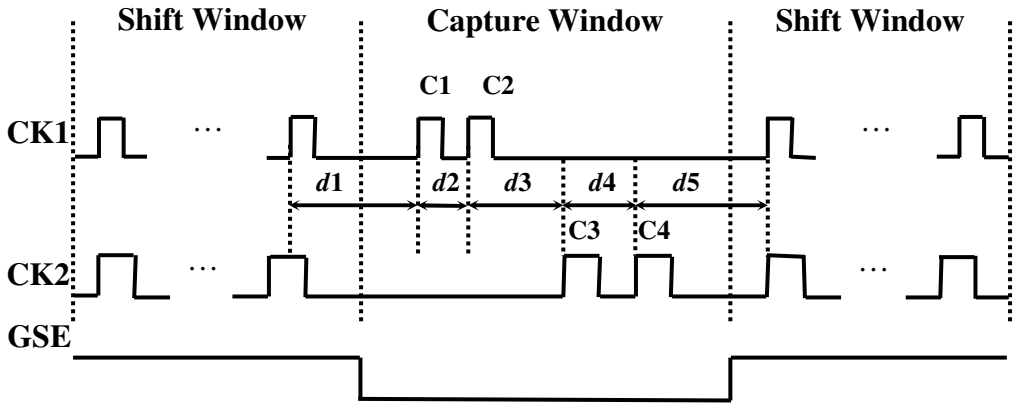
Aligned double-capture - I



Aligned double-capture - II

Staggered Double-Capture

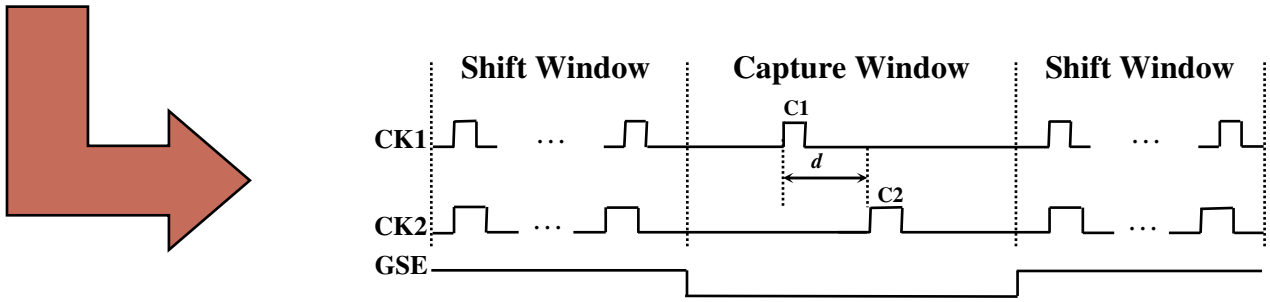
In the capture window, two capture pulses are generated for each clock domain. The first two capture pulses are used to create transitions at the outputs of scan cells, and the output responses to the transitions are caught by the next two capture pulses, respectively.



Staggered double-capture

Fault Detection

- Intra-clock-domain delay fault detection is relatively easy.
- Testing inter-clock-domain delay faults is more complex.
- A single capture yields the highest fault coverage of inter-clock-domain delay faults.



Fault Detection Capability

Capture-clocking scheme	Intra-structural	Intra-delay	Inter-structural	Inter-delay	Sync. design	Async. design
One-hot single-capture	√	-	√	-	√	√
Staggered single-capture	√	-	√	√	√	√
One-hot skewed-load	√	√	√	-	√	√
Aligned skewed-load	√	√	√	√	√	-
Staggered skewed-load	√	√	√	√	√	√
One-hot double-capture	√	√	√	-	√	√
Aligned double-capture	√	√	√	√	√	-
Staggered double-capture	√	√	√	√	√	√

Note: A hybrid double-capture scheme using staggered double-capture and aligned double-capture seems to be the preferred scheme for true at-speed testing

A Design Practice

An example of designing a logic BIST system for testing a scan-based design (core) comprising two clock domains using s38417 and s38584. The two clock domains are taken from the ISCAS-1989 benchmark circuits [Brglez 1989].

Clock Domain	*No. of PIs*	*No. of POs*	*No. of flip-flops*	*No. of gates*
CD1 (s38417)	28	106	1636	22179
CD2 (s38584)	12	278	1452	19253

Design statistics

Design flow

- BIST Rule Checking and Violation Repair
- Logic BIST System Design
- RTL BIST Synthesis
- Design Verification and Fault Coverage Enhancement

BIST Rule Checking and Violation Repair

All DFT rule violations of the scan design rules and BIST-specific design rules must be repaired. In addition, we should be aware of the following design parameters:

- The number of test clocks present in the design
- The number of set/reset clocks present in the design

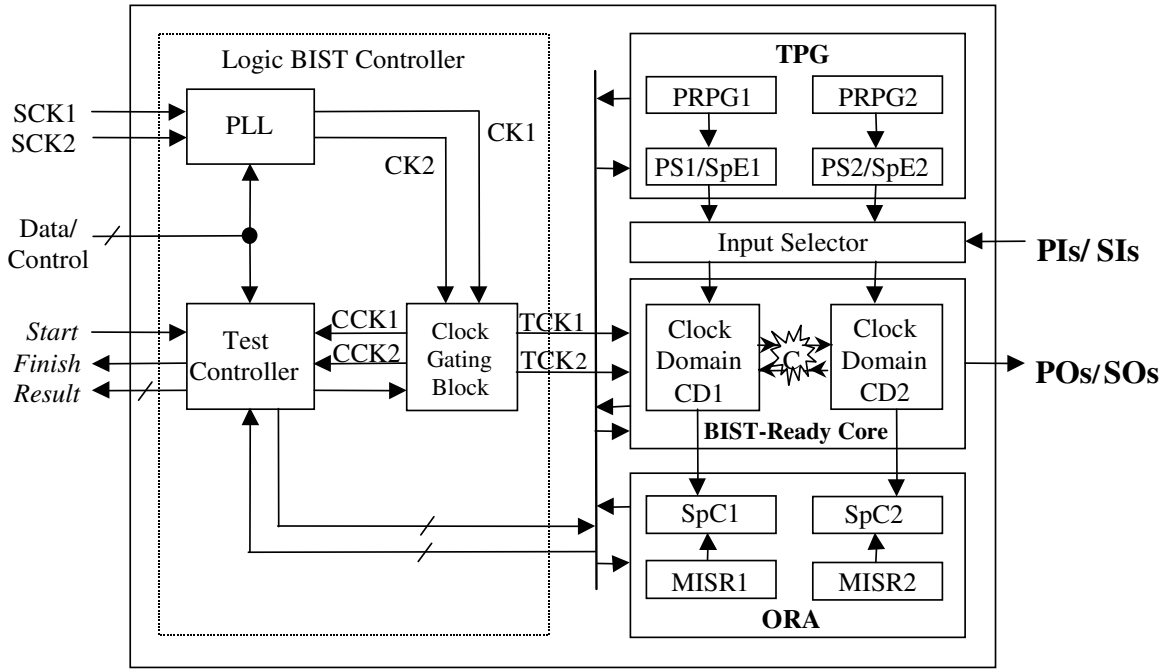
Logic BIST System Design

The second step is to design the logic BIST system at the RTL, including:

- The type of logic BIST architecture to adopt
- The number of PRPG-MISR (or PEPG-MISR) pairs to use
- The length of each PRPG-MISR (or PEPG-MISR) pair
- The faults to be tested and BIST timing control diagrams to be used
- The types of optional logic to be added

Logic BIST Architecture

We choose to implement a STUMPS-based architecture, since it is easy to integrate with scan/ATPG and is the industry widely used architecture.



A logic BIST system for testing a design with 2 cores

TPG and ORA

Next, we need to determine the length of each PRPG-MISR pair. Using a separate PRPG-MISR pair for each clock domain allows us to reduce the Length of each PRPG and MISR.

Clock Domain	*No. of scan chains*	*No. of shared SIs or SOs*	*Max. scan chain length*	*PRPG length*	*MISR length*
CD1 (s38417)	20	20	82	28	47
CD2 (s38584)	20	10	73	25	45

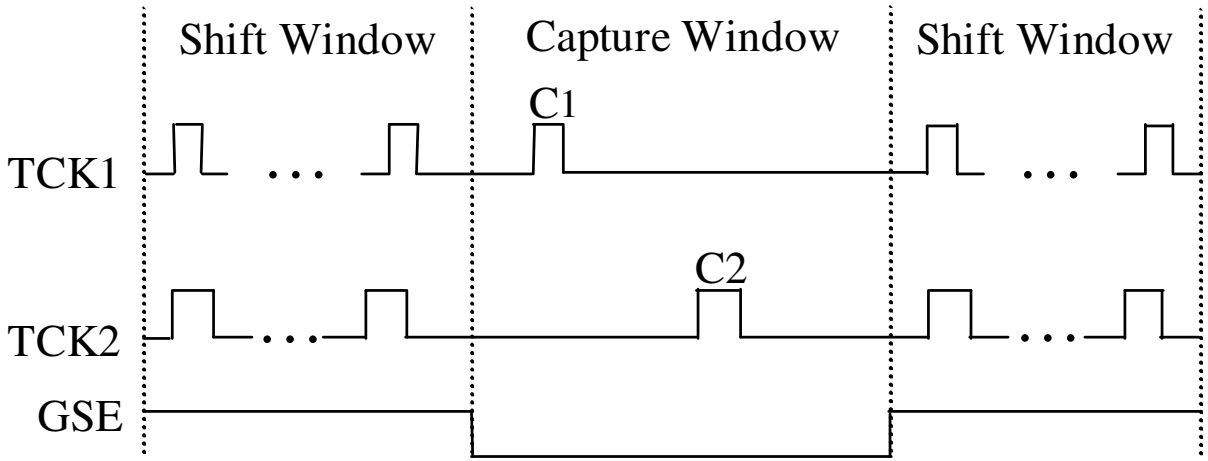
PRPG-MISR Choices

Test Controller

The test controller plays a central role in coordinating the overall BIST operation. Often, external signals are controlled through an IEEE 1149.1 Boundary-Scan Standard based test access port (TAP) controller.

Test Controller (cont)

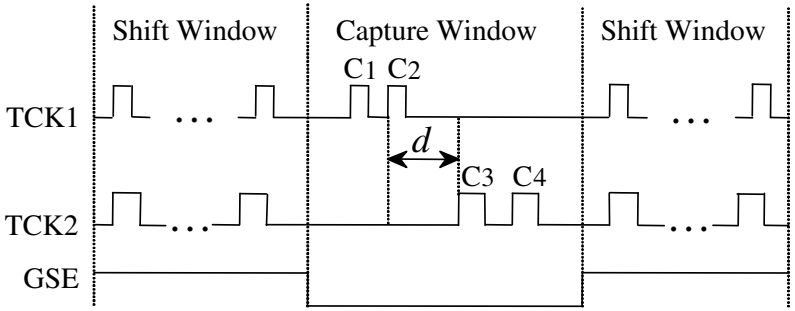
In order to test structural faults in the BIST-ready core, we choose the Staggered single-capture approach.



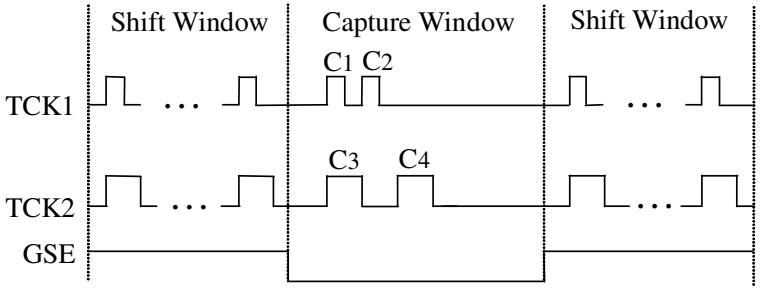
Slow-speed timing control using staggered single-capture

Test Controller (cont)

In order to test delay faults in the BIST-ready core, we choose the Staggered double-capture approach if CD1 and CD2 are asynchronous, or the aligned double-capture approach if they are synchronous.



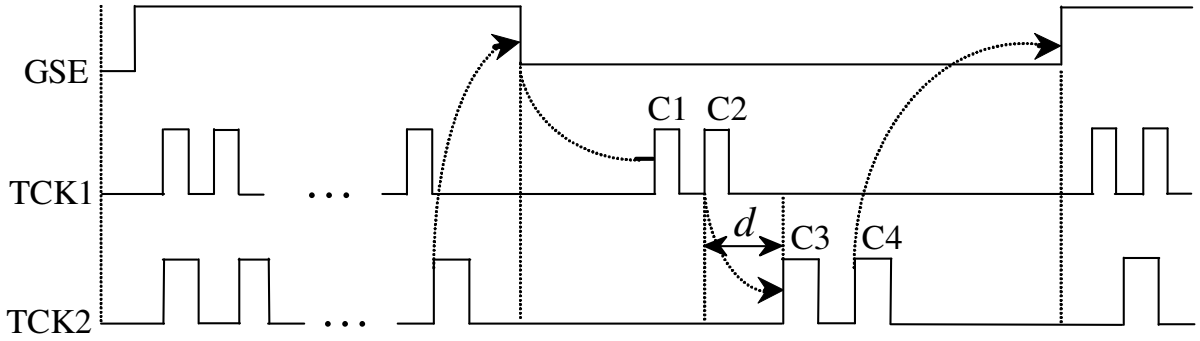
Staggered double-capture



Aligned double-capture

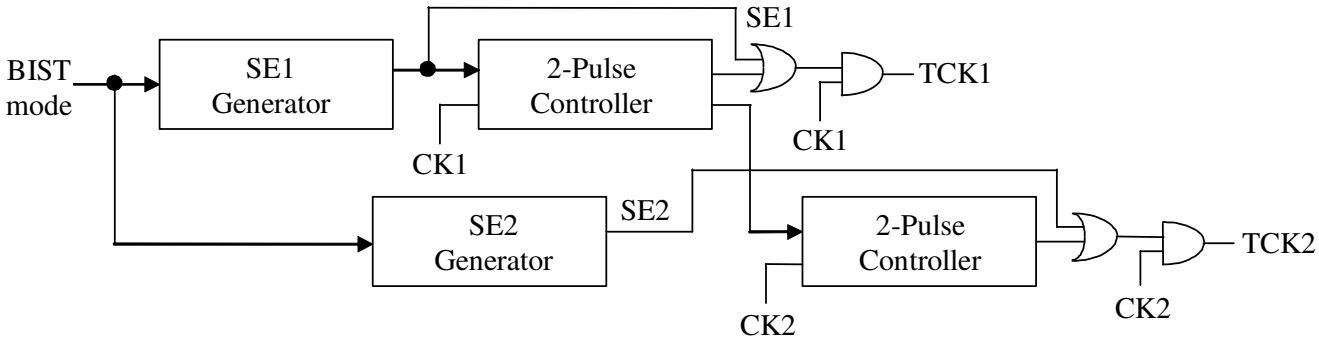
Clock Gating Block

In order to generate an ordered sequence of single-capture or double-capture clocks, clock suppression [Rajski 2003] [Wang 2004], daisy-chain clock-triggering, or token-ring clock-enabling [Wang 2005a] can be used.

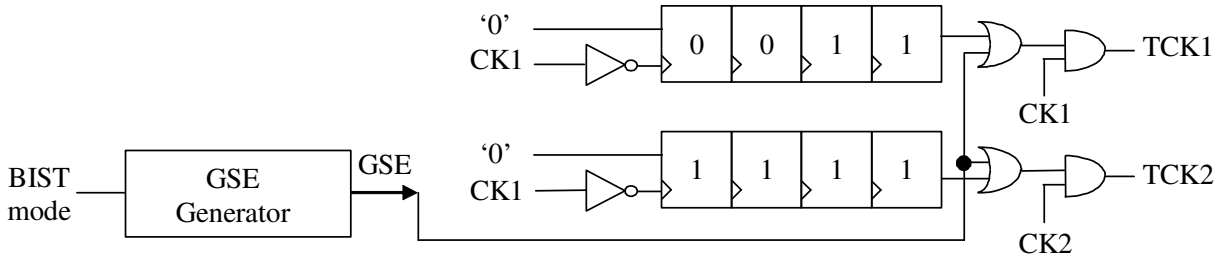


Daisy-chain clock-triggering

Clock Gating Block (cont)



A daisy-chain clock-triggering circuit



A clock suppression circuit

Re-Timing Logic

we recommend adding two pipelining registers between each PRPG and the BIST-ready core, and two additional pipelining registers between the BIST-ready core and each MISR. In this case, the maximum scan chain length for each clock domain,CD1 or CD2, is effectively increased by 2, not 4.

Fault Coverage Enhancing Logic and Diagnostic Logic

In order to improve the circuit's fault coverage, we recommend adding extra test points and additional logic for top-up ATPG support at the RTL.

We also recommend including diagnostic logic in the RTL BIST code to facilitate debug and diagnosis.

Test mode	*CD1 effective chain count*	*CD2 effective chain count*
Normal	0	0
BIST	20	20
ATPG	20	10
ATPG compression	20	20
Serial debug and diagnosis	1	1

Example test modes to be supported by the logic BIST system

RTL BIST Synthesis

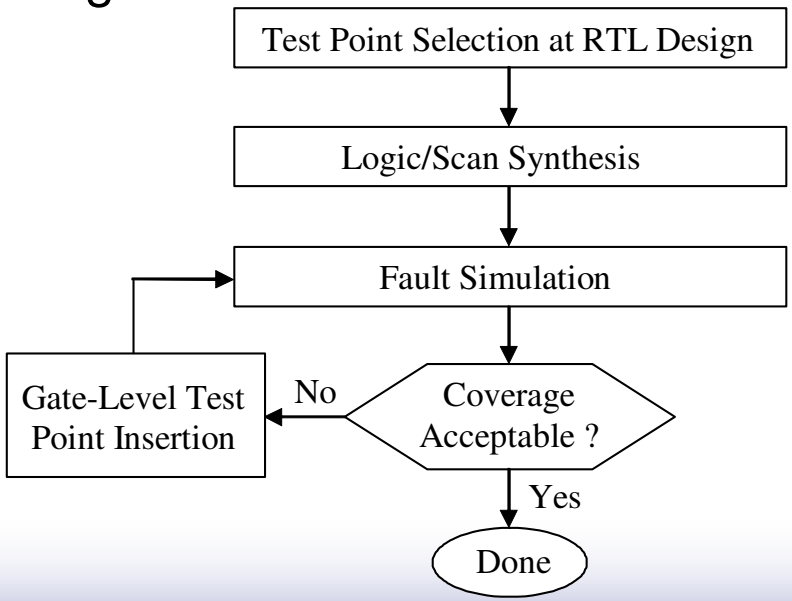
At this stage, it is possible to either design the logic BIST system by hand or generate the RTL code automatically using a (commercially available) RTL logic BIST tool.

In either case, the number of scan chains for each clock domain should be specified along with the names of their associated scan inputs (SIs) and scan outputs (SOs) without inserting the actual scan chains into the circuit.

Design Verification and Fault Coverage Enhancement

Finally, the synthesized netlist needs to be verified with functional and/or Timing verification.

Next, fault simulation needs to be performed on the pseudo-random Patterns generated by the TPG in order to determine the circuit's fault coverage.



Fault simulation and test point insertion flow

Concluding Remarks

- STUMPS is an industry widely adopted logic BIST architecture, but hits problems due to low fault coverage.
- Some challenges ahead
  - Whether the CBILBO-based architecture proposed by Wang and McCluskey would perform as it always guarantee 100% single stuck-fault coverage.
  - Whether pseudo-exhaustive testing would become the preferred BIST technique.