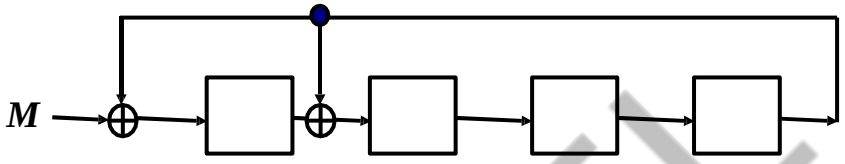
Lecture 27

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

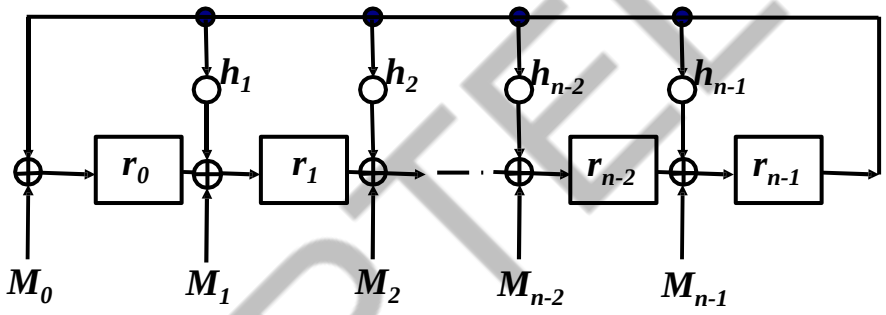
M''	r_0	r_1	r_2	r_3
1	**0**	**0**	**0**	**0**
0	1	0	0	0
1	0	1	0	0
1	1	0	1	0
0	1	1	0	1
0	1	0	1	0
1	0	1	0	1
1	0	1	1	0
R''	**1**	**0**	**1**	**1**

(c) Signature for fault f_2

A 4-stage SISR

Parallel Signature Analysis

Multiple-input signature register (MISR)

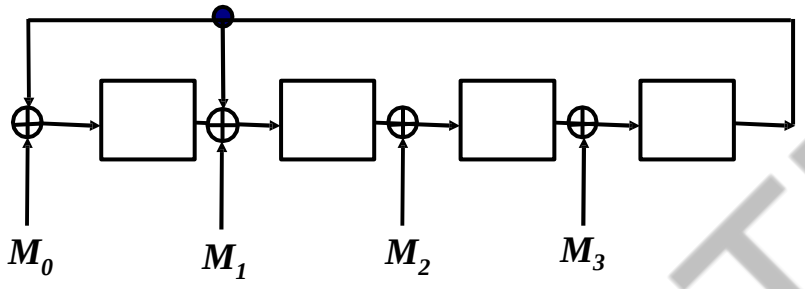


An n-input MISR can be remodeled as a single-input SISR with *effective input sequence M(x)* and *effective error polynomial E(x)*

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

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

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

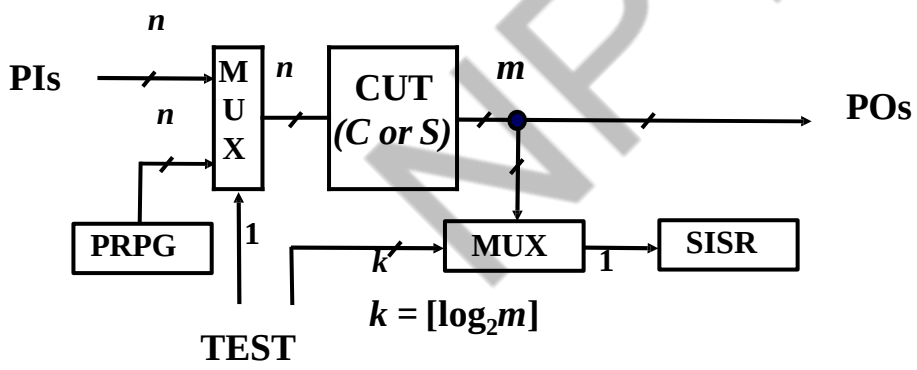
Four Types of BIST Architectures:

- 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]

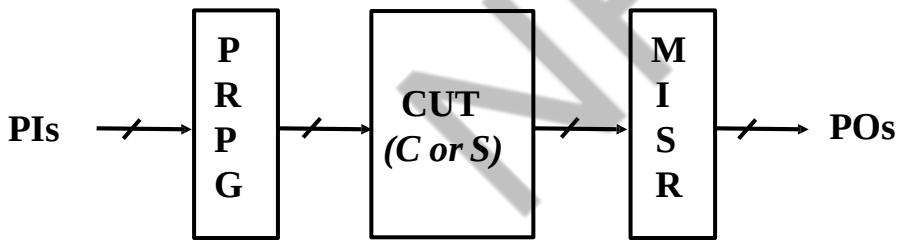


CSBL Architecture

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]

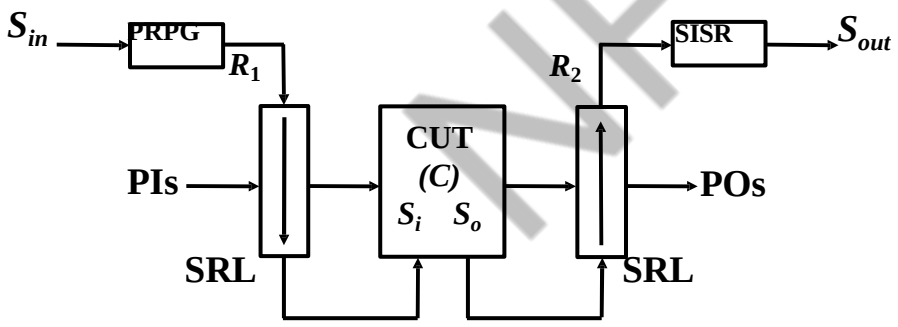


BEST Architecture

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]



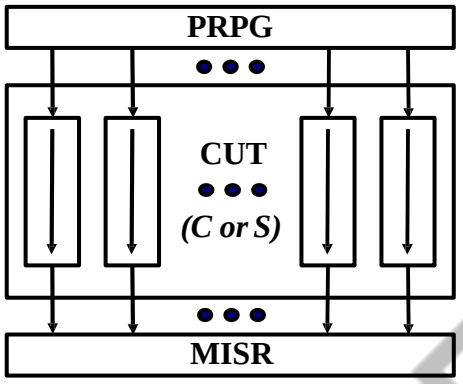
LOCST Architecture

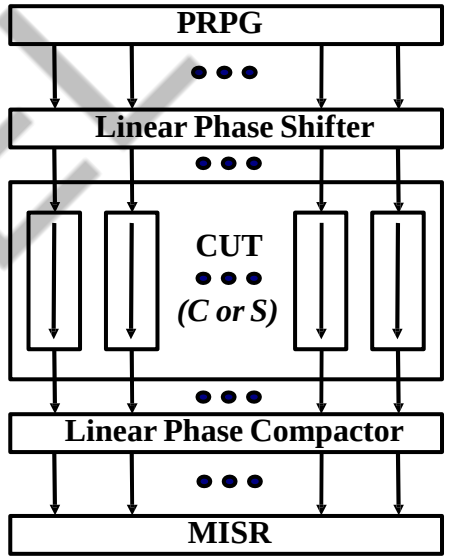
Type II - *Self-Testing Using MISR and Parallel SRSG (STUMPS)*

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***

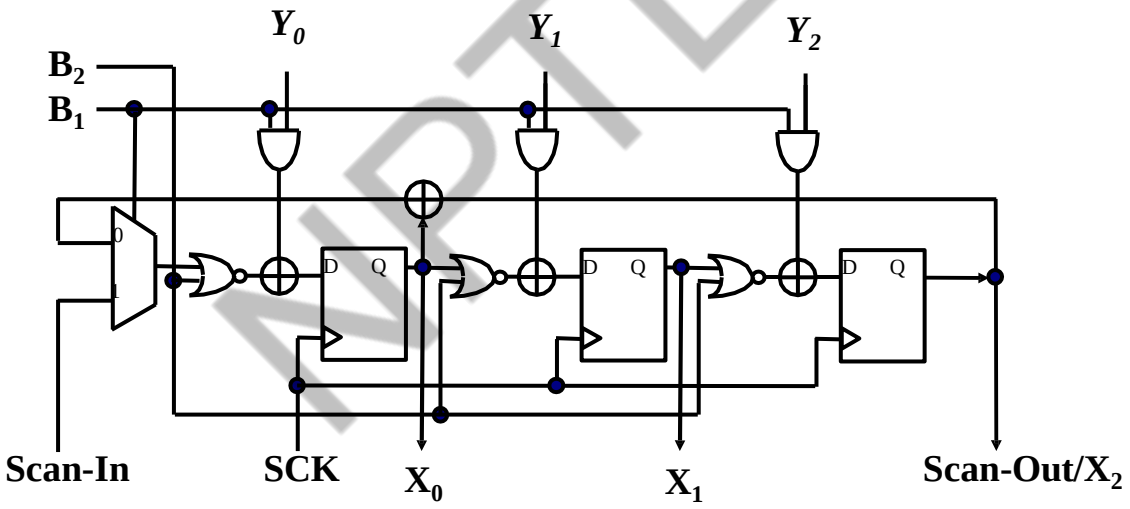
Type III - *Built-In Logic Block Observer (BILBO)*

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

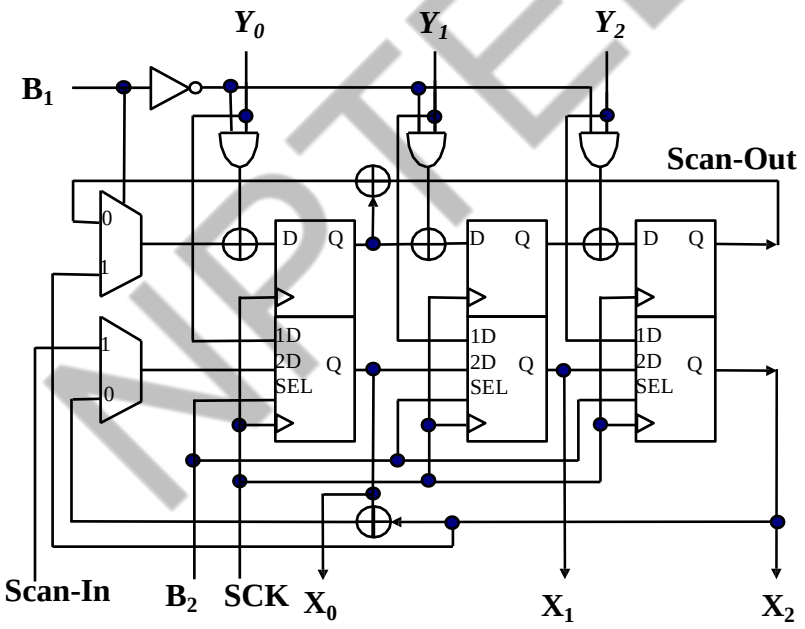
B_1	B_2	Operation mode
1	1	Normal
0	0	Scan
1	0	Mixed Test Generation and Signature Analysis
0	1	Reset



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



A 3-stage concurrent BILBO (CBILBO)

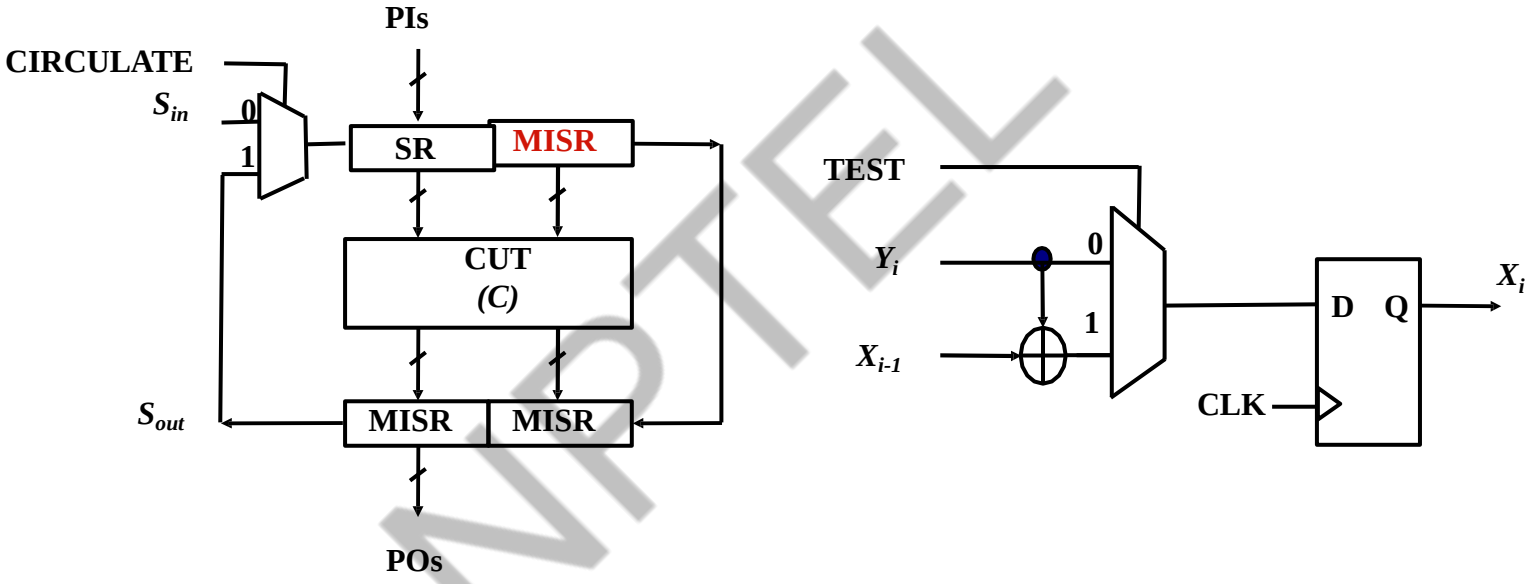
[Wang 1986c]

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]

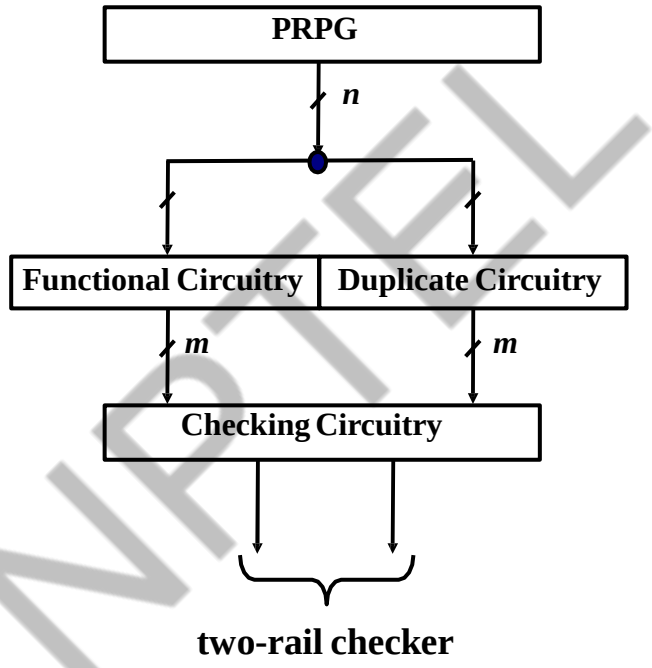
Circular Self-Test Path



(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

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.

Lecture 28

Test Compression

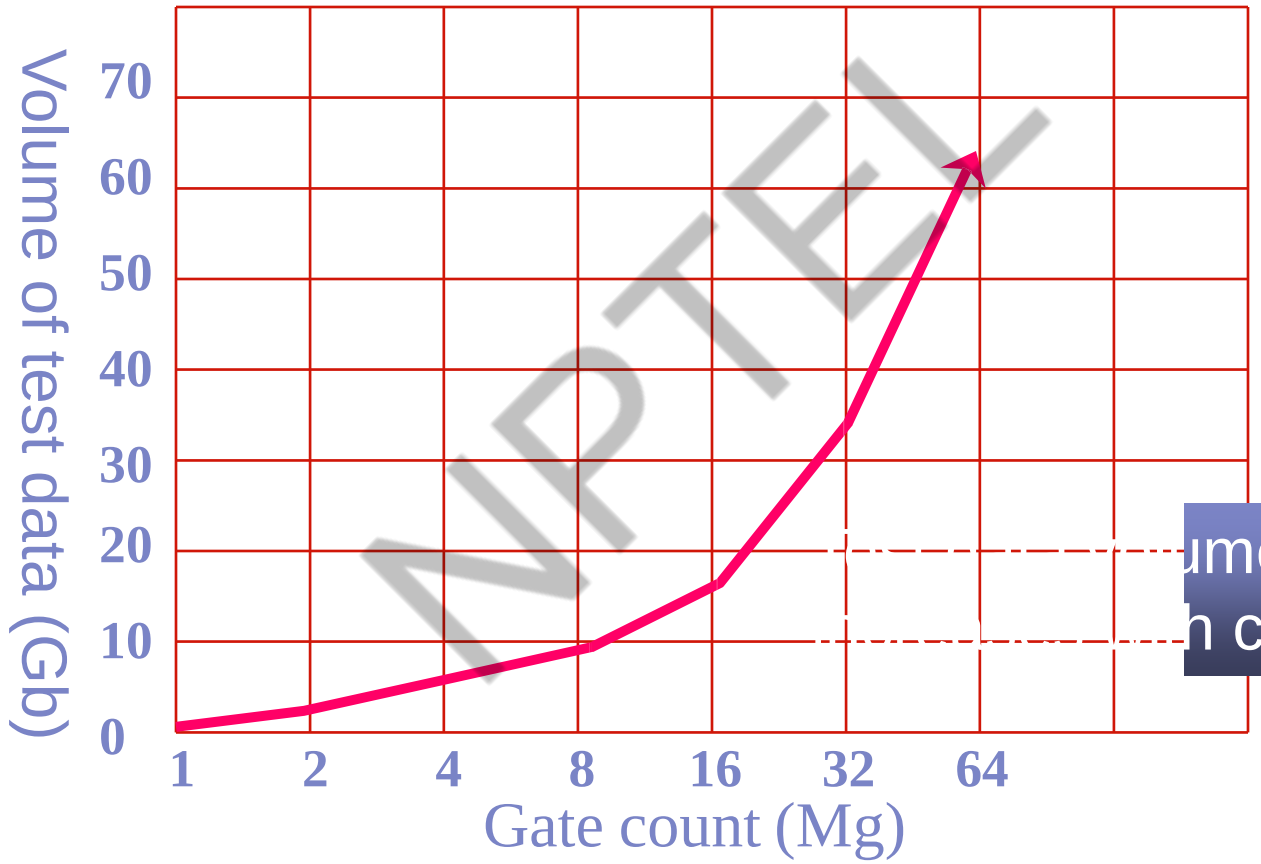
Test Compression

- Introduction
- Test Stimulus Compression
- Test Response Compaction
- Industry Practices
- Concluding Remarks

Introduction

- Why do we need test compression?
  - Test data volume
  - Test time
  - Test pins
- Why can we compress test data?
  - Deterministic test vector has “don’t care” (X’s)

Test data volume v.s. gate count



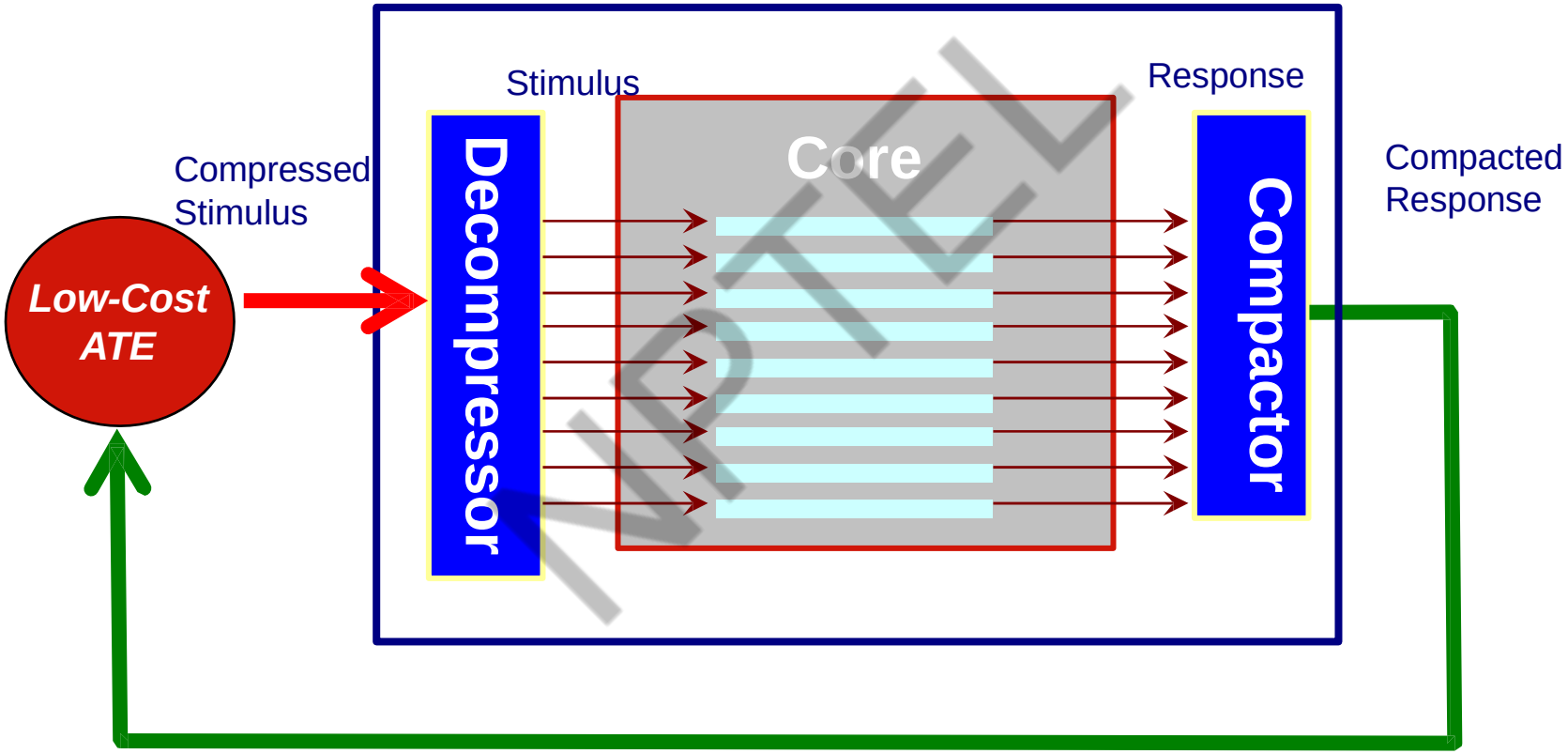
ume
h circuit size

(*Source: Blyler, Wireless System Design, 2001*)

Test compression categories

- Test Stimulus Compression
  - Code-based schemes
  - Linear-decompression-based schemes
  - Broadcast-scan-based schemes
- Test Response Compaction
  - Space compaction
  - Time compaction
  - Mixed time and space compaction

Architecture for test compression



Test stimulus compression

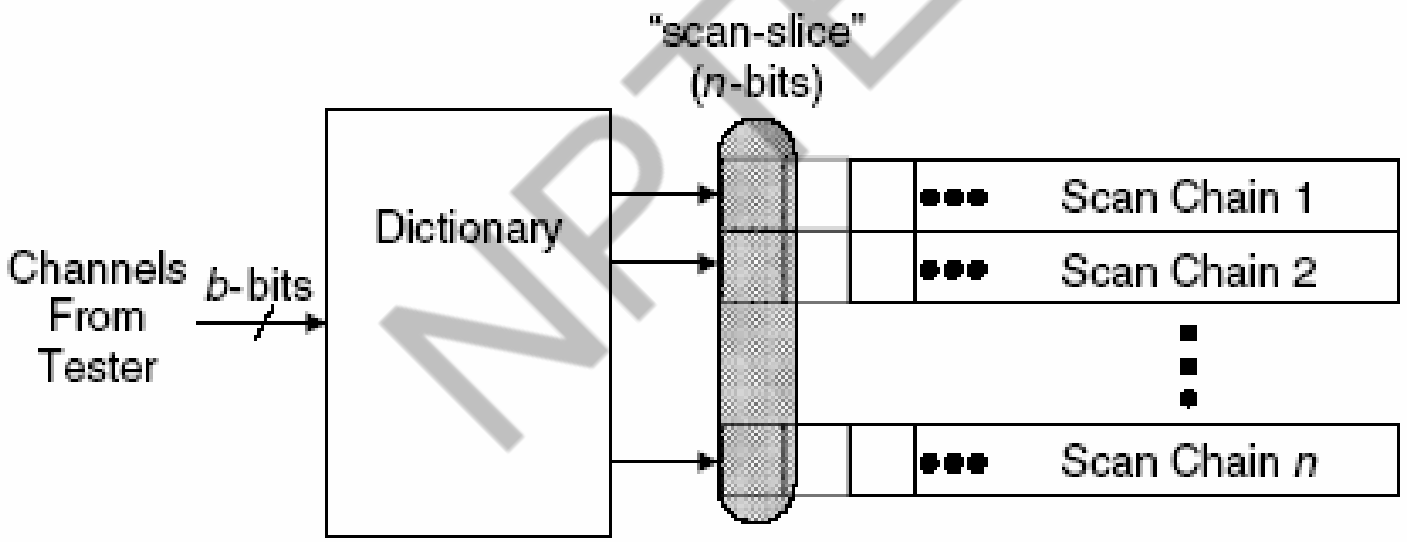
- Code-based schemes
- Linear-decompression-based schemes
- Broadcast-scan-based schemes

Test stimulus compression

- Code-based schemes
  - Dictionary code (fixed-to-fixed)
  - Huffman code (fixed-to-variable)
  - Run-length code (variable-to-fixed)
  - Golomb code (variable-to-variable)

Code-based schemes

- Dictionary code (fixed-to-fixed)



Lecture 29

Code-based schemes

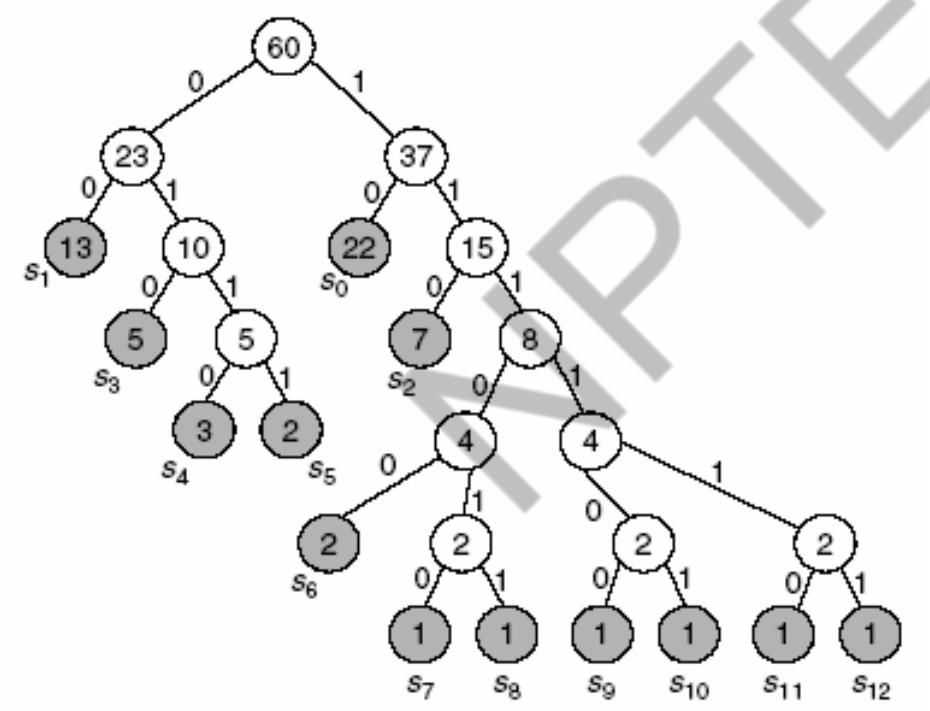
- Huffman code (fixed-to-variable)

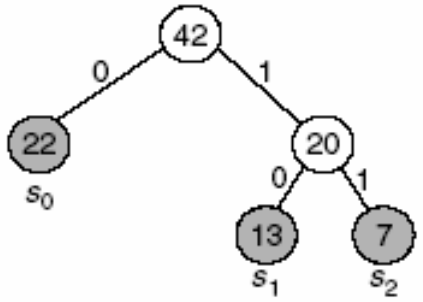
Symbol	Frequency	Pattern	Huffman Code	Selective Code
S_0	22	0010	10	10
S_1	13	0100	00	110
S_2	7	0110	110	111
S_3	5	0111	010	00111
S_4	3	0000	0110	00000
S_5	2	1000	0111	01000
S_6	2	0101	11100	00101
S_7	1	1011	111010	01011
S_8	1	1100	111011	01100
S_9	1	0001	111100	00001
S_{10}	1	1101	111101	01101
S_{11}	1	1111	111110	01111
S_{12}	1	0011	111111	00011
S_{13}	0	1110	—	—
S_{14}	0	1010	—	—
S_{15}	0	1001	—	—

0010 0100 0010 0110 0000 0010 1011 0100 0010 0100 0110 0010
0010 0100 0010 0110 0000 0110 0010 0100 0110 0010 0010 0000
0010 0110 0010 0010 0010 0100 0100 0110 0010 0010 1000 0101
0001 0100 0010 0111 0010 0010 0111 0111 0100 0100 1000 0101
1100 0100 0100 0111 0010 0010 0111 1101 0010 0100 1111 0011

Code-based schemes

- Huffman code (fixed-to-variable)

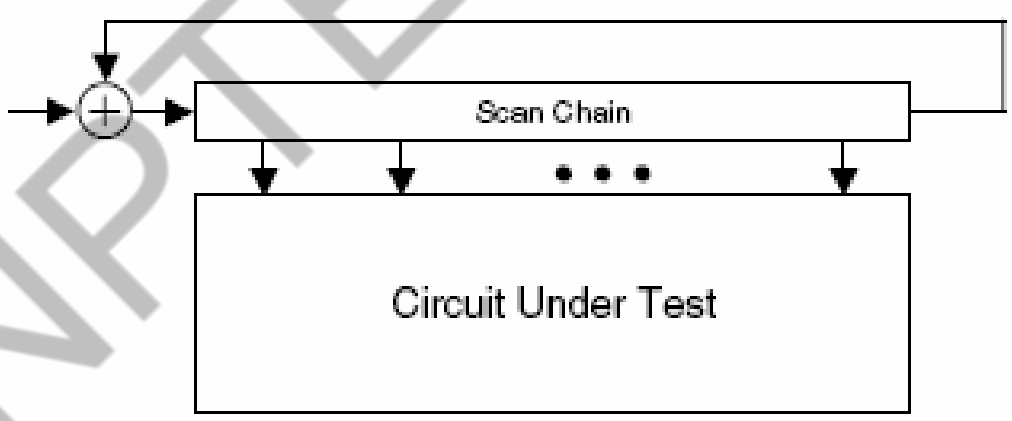




Code-based schemes

- Run-length code (variable-to-fixed)

Symbol	Codeword
1	000
01	001
001	010
0001	011
00001	100
000001	101
0000001	110
0000000	111



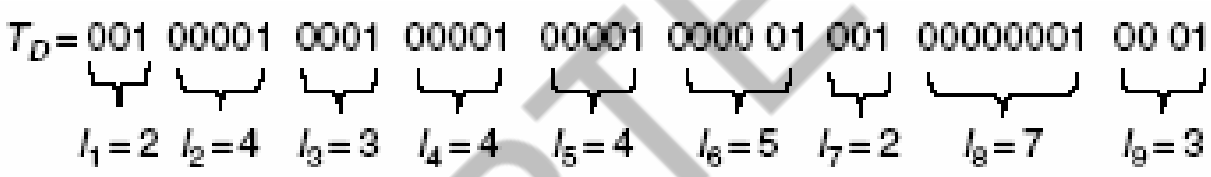
Code-based schemes

- Golomb code (variable-to-variable)

Group	Run-Length	Group Prefix	Tail	Codeword
A_1	0	0	0 0	0 0 0
	1		0 1	0 0 1
	2		1 0	0 1 0
	3		1 1	0 1 1
A_2	4	1 0	0 0	1 0 0 0
	5		0 1	1 0 0 1
	6		1 0	1 0 1 0
	7		1 1	1 0 1 1
A_3	8	1 1 0	0 0	1 1 0 0 0
	9		0 1	1 1 0 0 1
	10		1 0	1 1 0 1 0
	11		1 1	1 1 0 1 1
...	...	...	...	...

Code-based schemes

- Golomb code (variable-to-variable)



Using Golomb code shown in Table 6.4

T_E = 010 1000 011 1000 1000 1001 010 1011 011

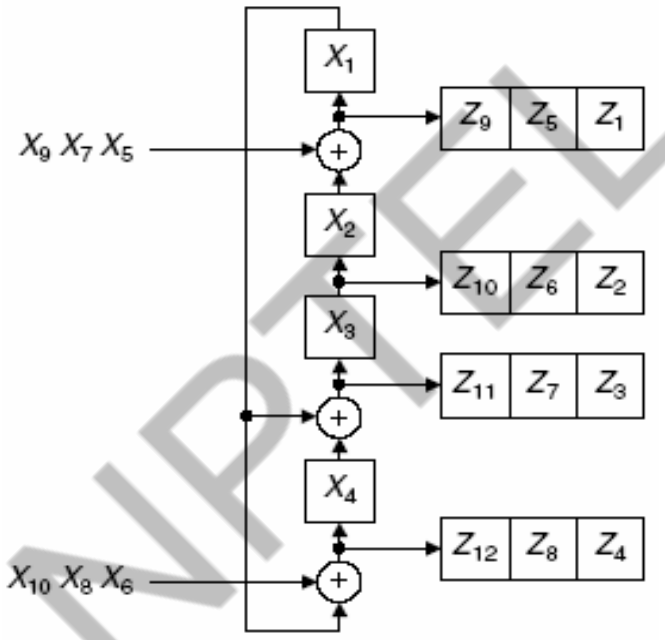
The length of T_D is 43 bits

The length of T_E is 32 bits

Test stimulus compression

- Linear-decompression-based schemes
  - Combinational linear decompressors
  - Fixed-length sequential linear decompressors
  - Variable-length sequential linear decompressors
  - Combined linear and nonlinear decompressors

Linear-decompression-based schemes



$$Z_9 = X_1 \oplus X_4 \oplus X_9$$
$$Z_{10} = X_1 \oplus X_2 \oplus X_5 \oplus X_6$$
$$Z_{11} = X_2 \oplus X_3 \oplus X_5 \oplus X_7 \oplus X_8$$
$$Z_{12} = X_3 \oplus X_7 \oplus X_{10}$$

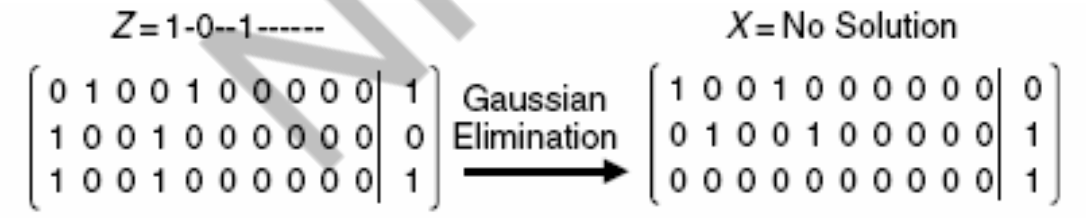
$$Z_5 = X_3 \oplus X_7$$
$$Z_6 = X_1 \oplus X_4$$
$$Z_7 = X_1 \oplus X_2 \oplus X_5 \oplus X_6$$
$$Z_8 = X_2 \oplus X_5 \oplus X_8$$

$$Z_1 = X_2 \oplus X_5$$
$$Z_2 = X_3$$
$$Z_3 = X_1 \oplus X_4$$
$$Z_4 = X_1 \oplus X_6$$

Linear-decompression-based schemes

$$\begin{pmatrix} 0&1&0&0&1&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0 \\ 1&0&0&1&0&0&0&0&0&0 \\ 1&0&0&0&0&1&0&0&0&0 \\ 0&0&1&0&0&0&1&0&0&0 \\ 1&0&0&1&0&0&0&0&0&0 \\ 1&1&0&0&1&1&0&0&0&0 \\ 0&1&0&0&1&0&0&1&0&0 \\ 1&0&0&1&0&0&0&0&1&0 \\ 1&1&0&0&1&1&0&0&0&0 \\ 0&1&1&0&1&0&1&1&0&0 \\ 0&0&1&0&0&0&1&0&0&1 \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \\ X_8 \\ X_9 \\ X_{10} \end{pmatrix} = \begin{pmatrix} Z_1 \\ Z_2 \\ Z_3 \\ Z_4 \\ Z_5 \\ Z_6 \\ Z_7 \\ Z_8 \\ Z_9 \\ Z_{10} \\ Z_{11} \\ Z_{12} \end{pmatrix}$$

Linear-decompression-based schemes

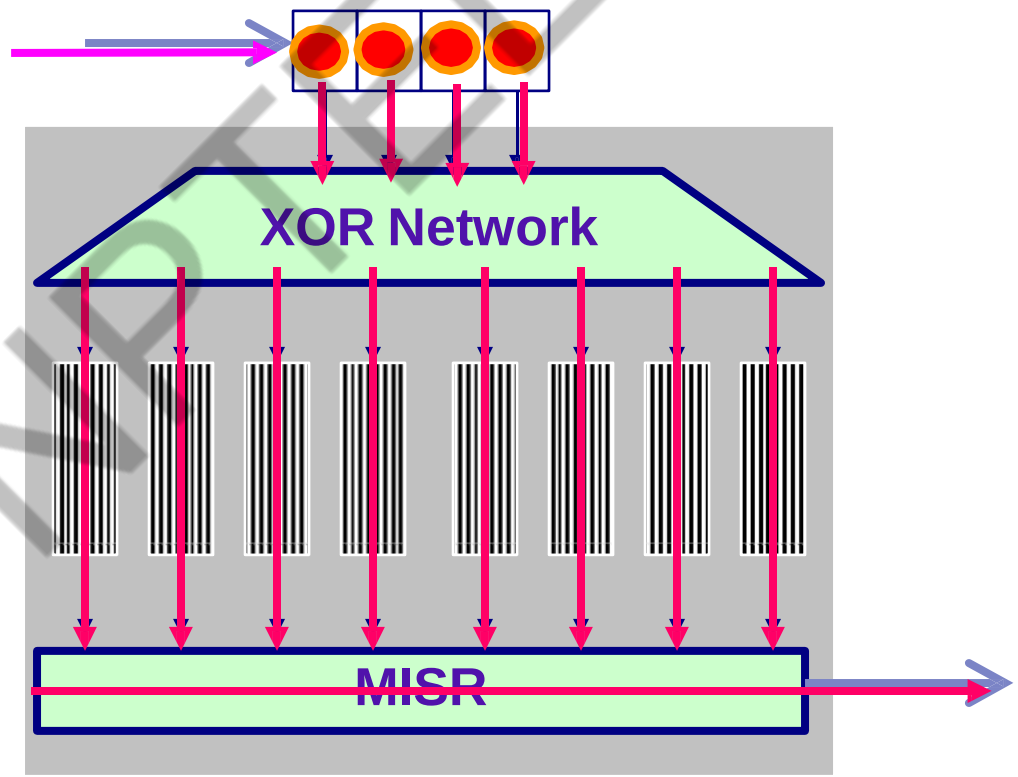


Lecture 30

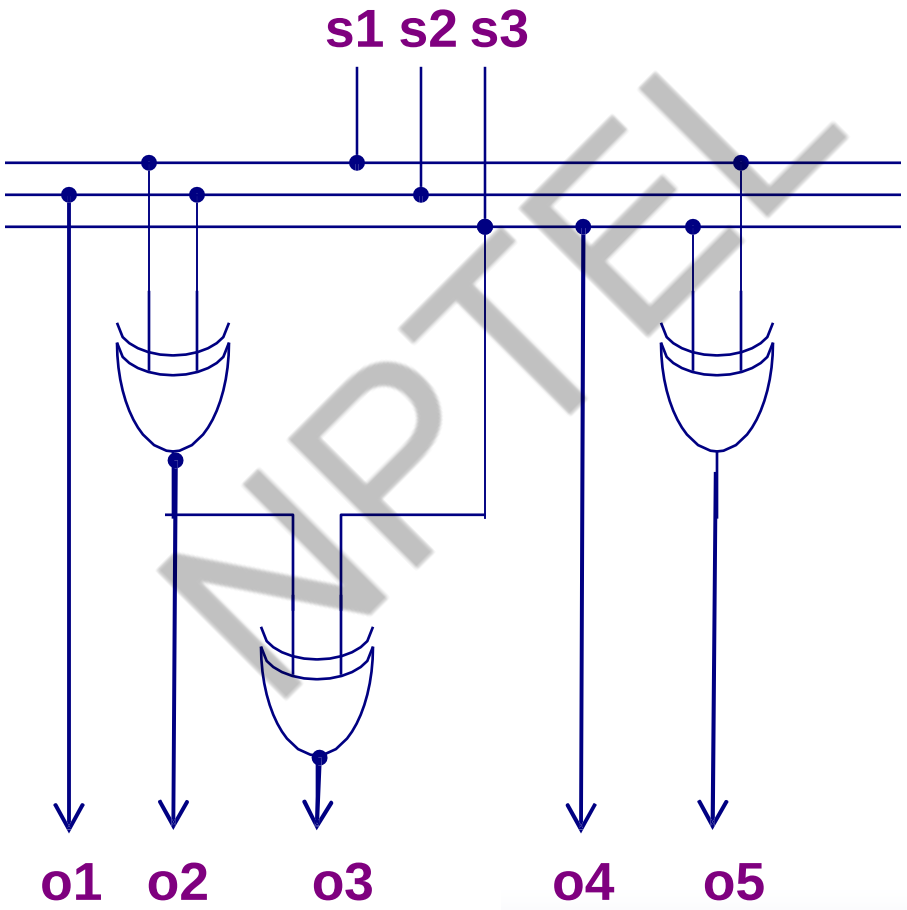
Linear-decompression-based schemes

- Combinational linear decompressors

XOR Network

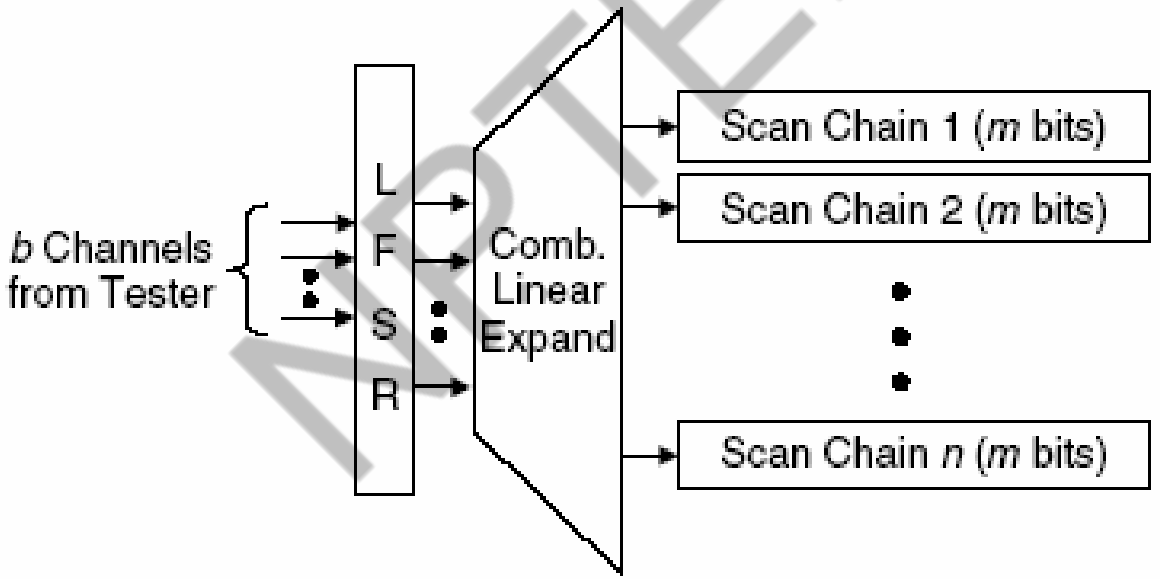


XOR network: a 3-to-5 example



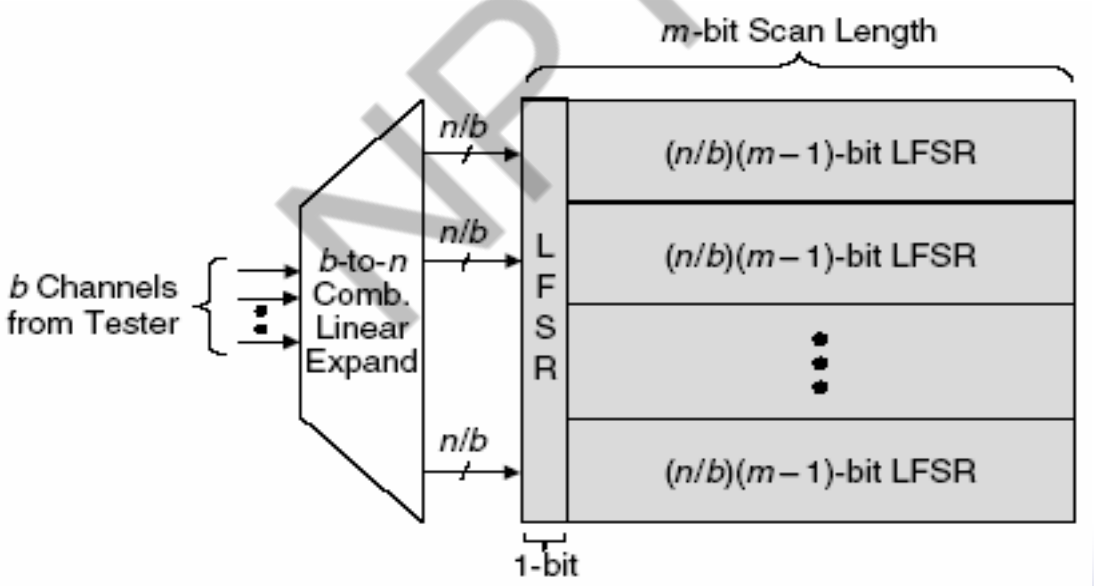
Linear-decompression-based schemes

- Fixed-length sequential linear decompressors



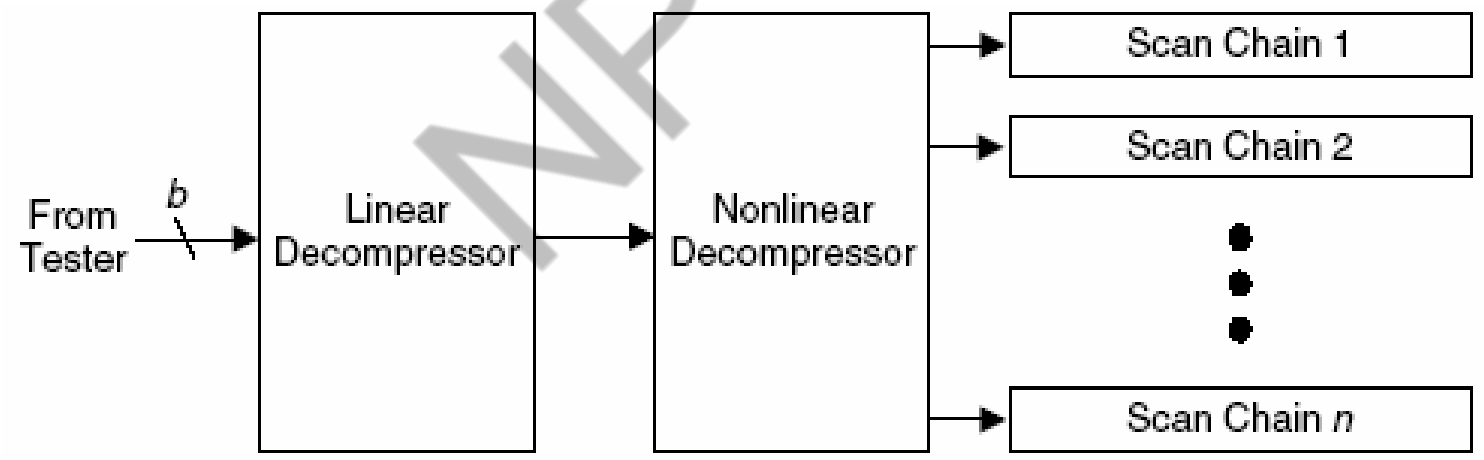
Linear-decompression-based schemes

- Variable-length sequential linear decompressors
  - Can vary the number of free variables
  - Better encoding efficiency
  - More control logic and control information



Linear-decompression-based schemes

- Combined linear and nonlinear decompressors
  - Specified bits tend to be highly correlated
  - Combine linear and nonlinear decompression together can achieve greater compression than either alone

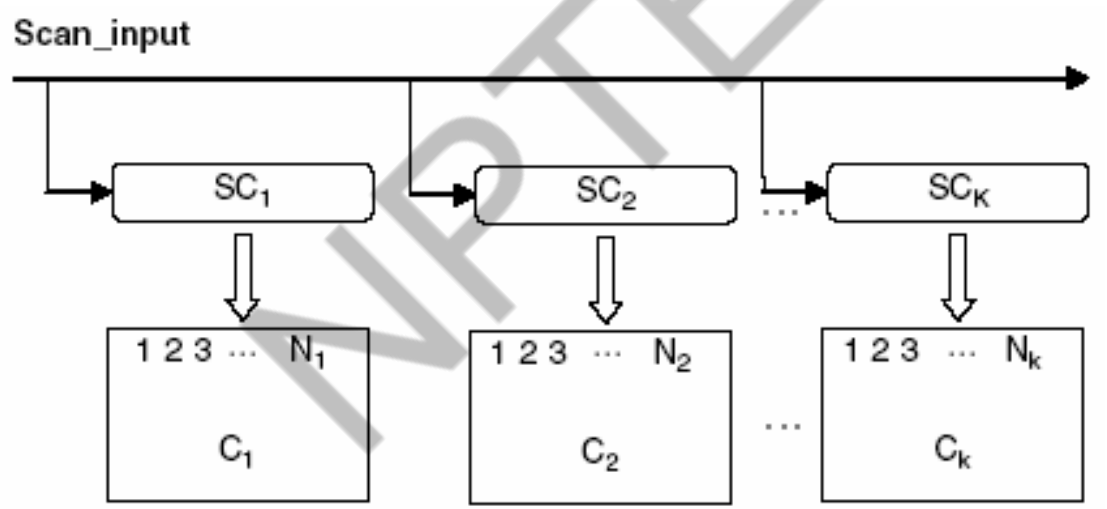


Test stimulus compression

- Broadcast-scan-based schemes
  - Broadcast scan
  - Illinois scan
  - Multiple-input broadcast scan
  - Reconfigurable broadcast scan
  - Virtual scan

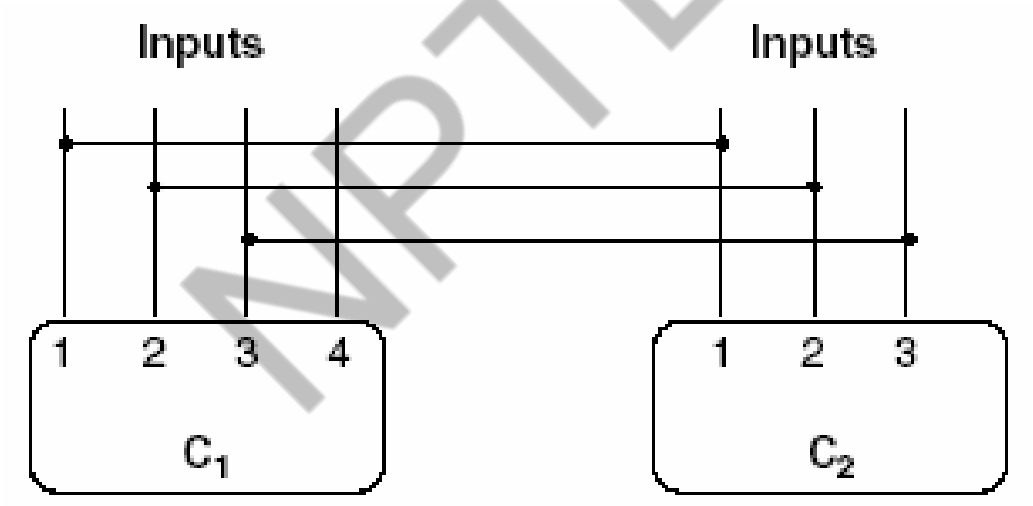
Broadcast-scan-based schemes

- Broadcast scan



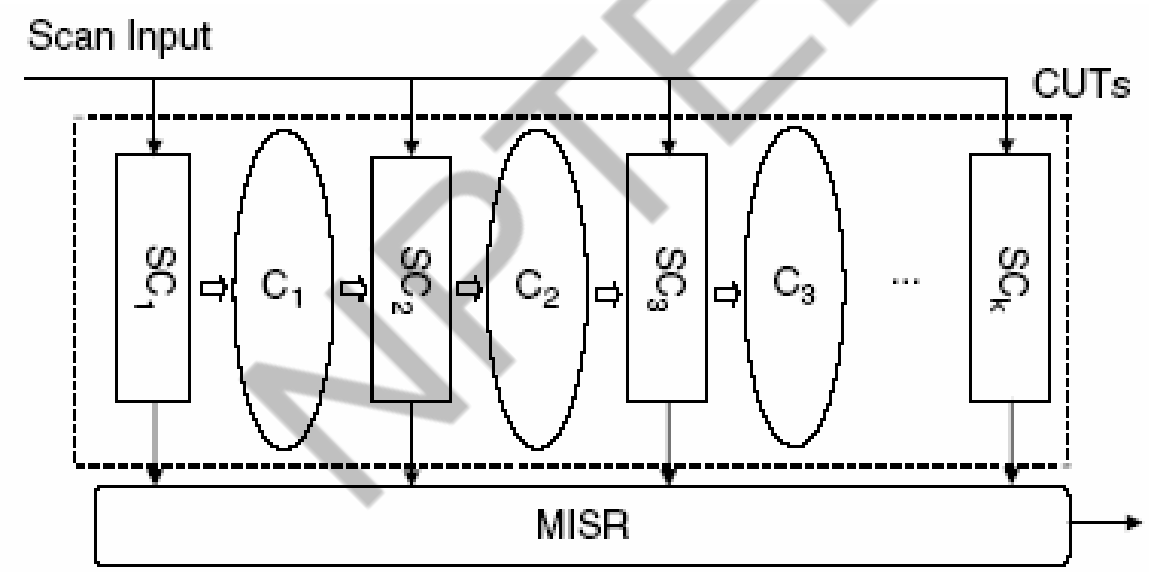
Generate patterns for broadcast scan

- Force ATPG tool to generate patterns for broadcast scan



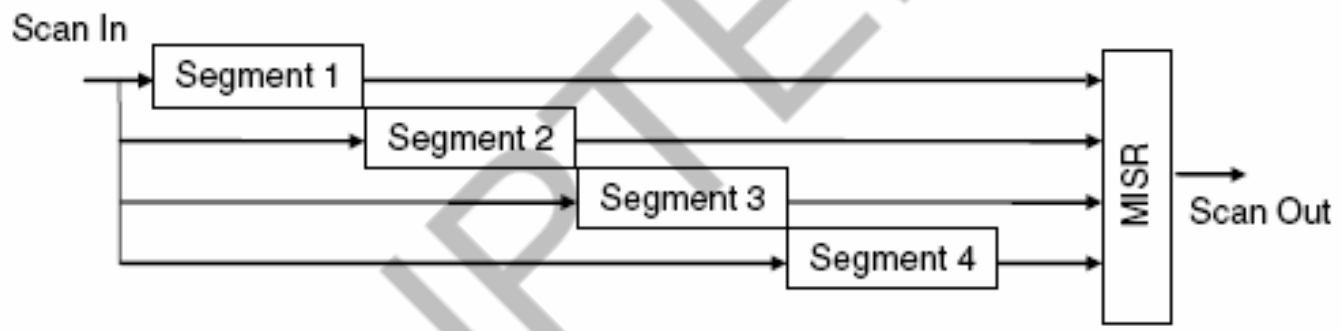
Broadcast scan for a pipelined circuit

- Broadcast scan for a pipelined circuit

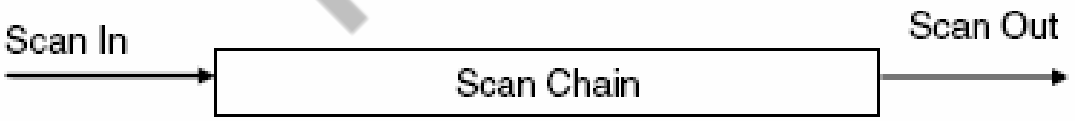


Broadcast-scan-based schemes

- Illinois scan architecture



(a) Broadcast mode



(b) Serial chain mode

Broadcast-scan-based schemes

- Reconfigurable broadcast scan
  - Reduce the number of channels that are required
  - Static reconfiguration
    - The reconfiguration can only be done when a new pattern is to be applied
  - Dynamic reconfiguration
    - The configuration can be changed while scanning in a pattern

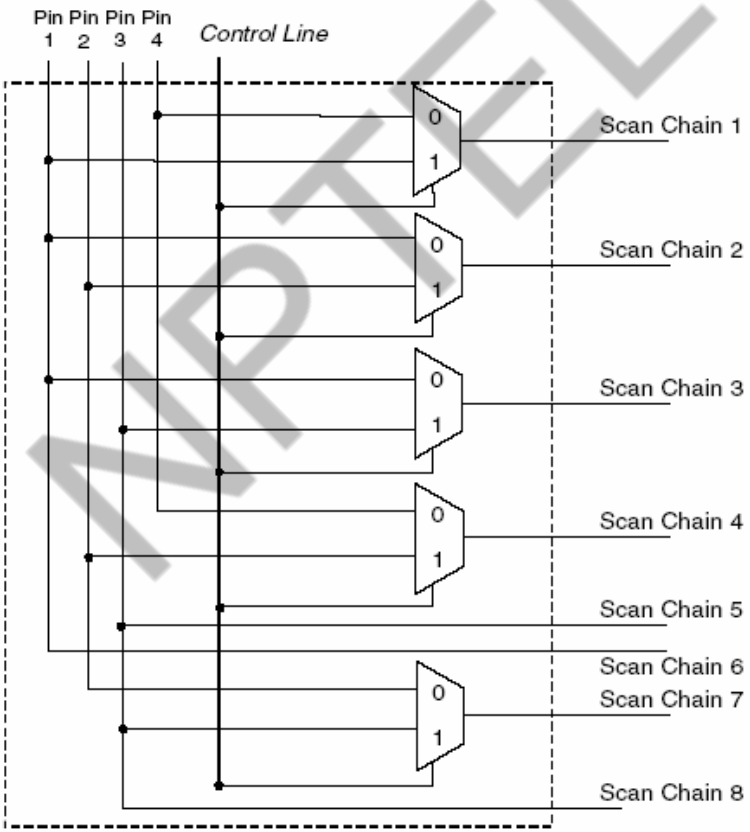
Broadcast-scan-based schemes

- First configuration is: 1->{2,3,6}, 2->{7}, 3->{5,8}, 4->{1,4}
- Other configuration is: 1->{1,6}, 2->{2,4}, 3->{3,5,7,8}

Scan Chain 1	1	X	1	X	X	X	0	0	X	X
Scan Chain 2	X	X	0	X	1	0	X	1	X	1
Scan Chain 3	X	X	X	X	1	1	1	X	X	1
Scan Chain 4	1	1	X	X	0	0	0	X	0	1
Scan Chain 5	0	X	1	X	X	X	X	X	X	X
Scan Chain 6	X	0	X	1	X	0	X	0	0	X
Scan Chain 7	0	X	0	X	X	1	1	X	X	X
Scan Chain 8	X	X	1	X	X	X	X	1	X	X

Broadcast-scan-based schemes

- Block diagram of MUX network



Broadcast-scan-based schemes

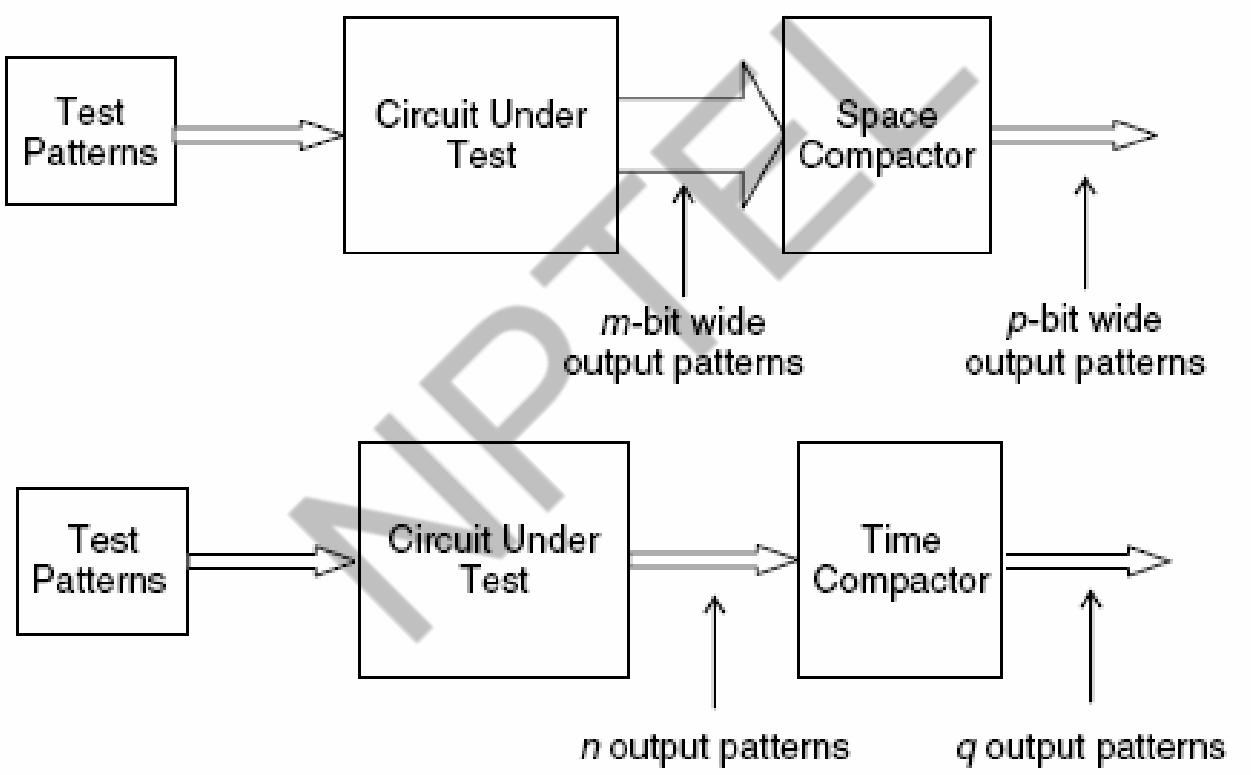
- Virtual scan
  - Pure MUX and XOR networks are allowed
  - No need to solve linear equations
  - Dynamic compaction can be effectively utilized during the ATPG process
  - Very little or no fault coverage loss

Test response compaction

- Space compaction
- Time compaction
- Mixed time and space compaction

Lecture 31

Test response compaction



Taxonomy of various response compaction schemes

Compaction Schemes	I		II		III	
	Space	Time	CFS	CFI	Linearity	Nonlinearity
Zero-aliasing Compactor [Chakrabarty 1998] [Pouya 1998]	√		√			√
Parity Tree [Karpovsky 1987]	√			√	√	
Enhanced Parity Tree [Sinanoglu 2003]	√	√	√		√	
X-Compact [Mitra 2004]	√			√	√	
q-Compactor [Han 2003]	√	√		√	√	
Convolutional Compactor [Rajski 2005]	√	√		√	√	
OPMISR [Barnhart 2002]	√	√		√	√	
Block Compactor [Wang 2003]	√	√		√	√	
i-Compact [Patel 2003]	√			√	√	
Compactor for SA [Wohl 2001]	√	√		√	√	
Scalable Selector [Wohl 2004]	√			√		√

Test response compaction

- Space compaction
  - Zero-aliasing linear compaction
  - X-compact
  - X-blocking
  - X-masking
  - X-impact

Space compaction

- Zero-aliasing linear compaction

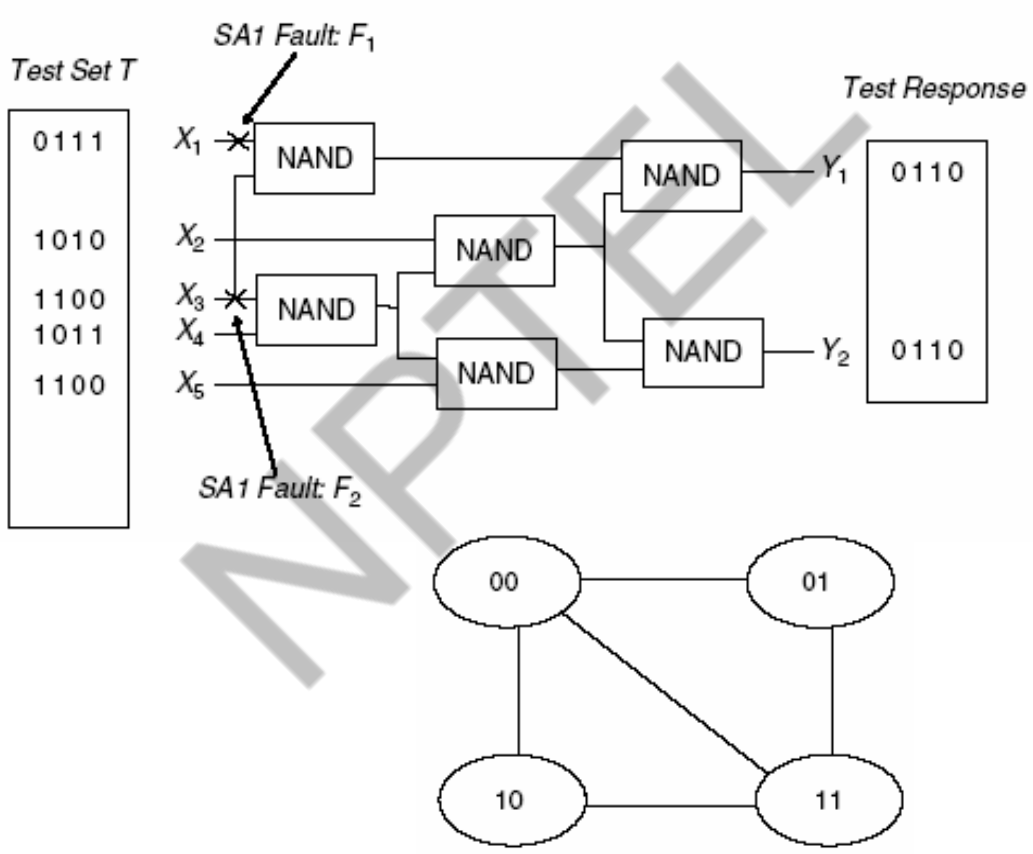
Theorem 6.1

For any test set T, for a circuit that implements function C, there exists a zero-aliasing output space compactor for C with q outputs where $q = \lceil \log_2(|T|+1) \rceil$.

Theorem 6.2

Let G be a response graph. If G is 2^q *colorable*, then there exists a q-output zero-aliasing space compactor for the circuit C.

An example of response graph



Space compaction

- X-compact
  - X-tolerant response compaction technique
  - X-compact matrix
  - Error masking

Space compaction

D X-compact

Theorem 6.3

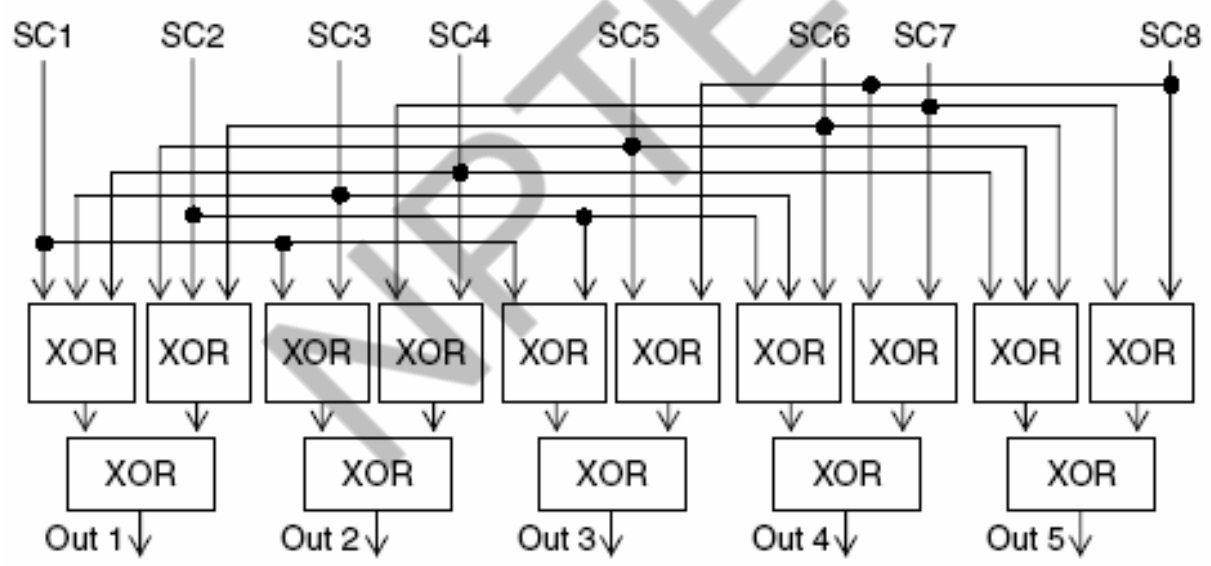
If only a single scan chain produces an error at any scan-out cycle, the X-compactor is guaranteed to produce errors at the X-compactor outputs at that scan-out cycle, if and only if no row of the X-compact matrix contains all 0's.

Theorem 6.4

Errors from any one, two, or an odd number of scan chains at the same scan-out cycle are guaranteed to produce errors at the X-compactor outputs at that scan-out cycle, if every row of the X-compact matrix is nonzero, distinct, and contains an odd number of 1's.

Space compaction

- X-compactor with 8 inputs and 5 outputs



X-compact Matrix

$$S=\begin{bmatrix} SC1 \\ SC2 \\ SC3 \\ SC4 \\ SC5 \\ SC6 \\ SC7 \\ SC8 \end{bmatrix}$$

$$M=\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

$$O=\begin{bmatrix} O1 \\ O2 \\ O3 \\ O4 \\ O5 \end{bmatrix}$$

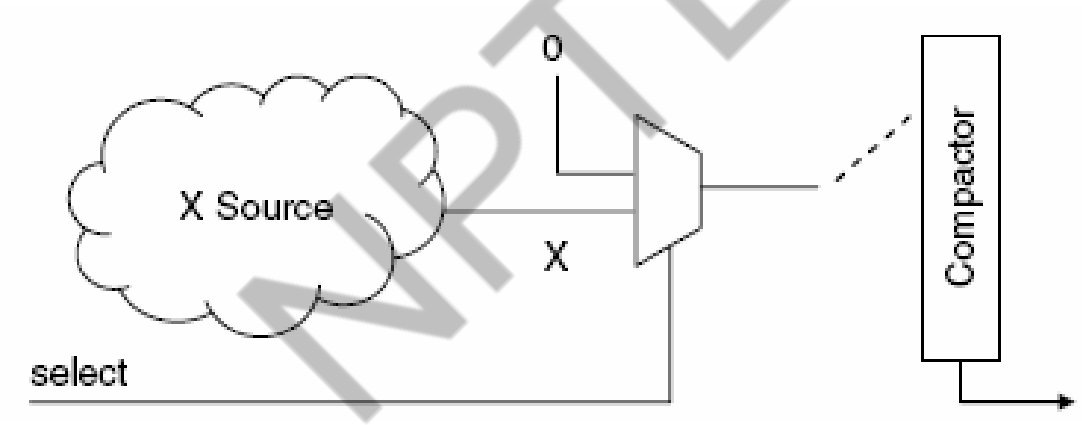
$$M^T \times S = O$$

Space compaction

- X-blocking (or X-bounding)
  - X's can be blocked before reaching the response compactor
  - Can ensure that no X's will be observed
  - May result in fault coverage loss
  - Add area overhead and may impact delay

Space compaction

- Illustration of the x-blocking scheme

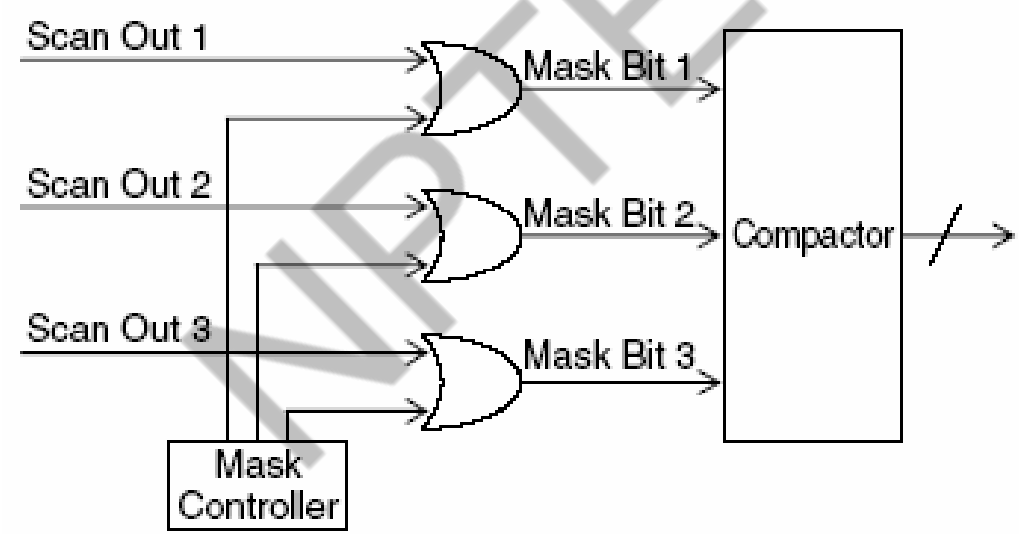


Space compaction

- X-masking
  - X's can be masked off right before the response compactor
  - Mask data is required to indicate when the masking should take place
  - Mask date can be compressed
    - Possible compression techniques are *weighted pseudo-random LFSR reseeding* or *run-length encoding*

Space compaction

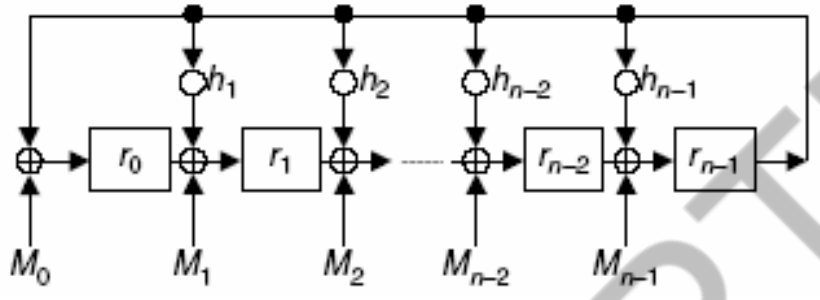
- An example of X-masking circuit



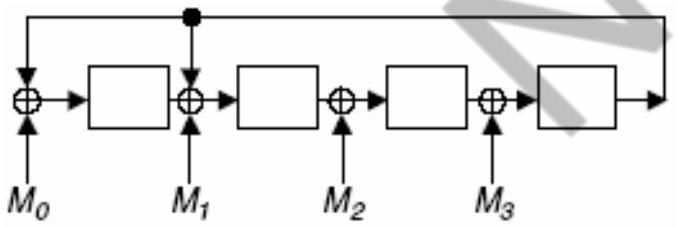
Test response compaction

- Time compaction
  - A time compactor uses sequential logic to compact test responses
  - MISR is most widely adopted
  - n-stage MISR can be described by specifying a *characteristic polynomial* of degree *n*

Multiple-input signature register (MISR)



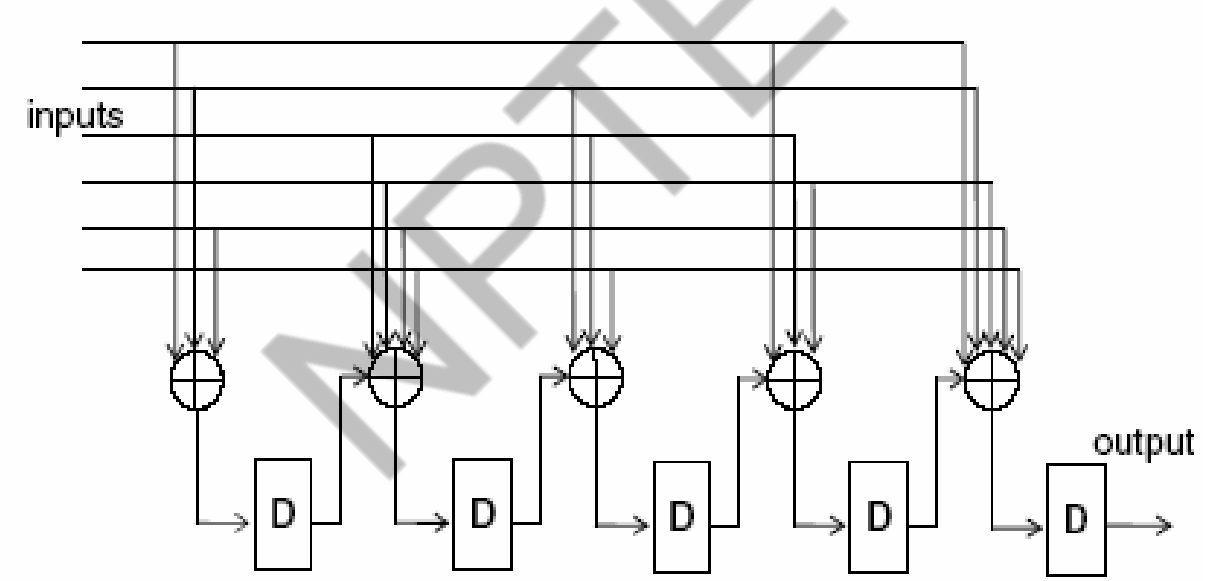
$$f(x) = 1 + h_1 x + h_2 x^2 + \cdots + h_{n-1} x^{n-1} + x^n$$



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

Test response compaction

- Mixed time and space compaction



Industry practices

- OPMISR+
- Embedded Deterministic Test
- Virtual Scan and UltraScan
- Adaptive Scan
- ETCompression

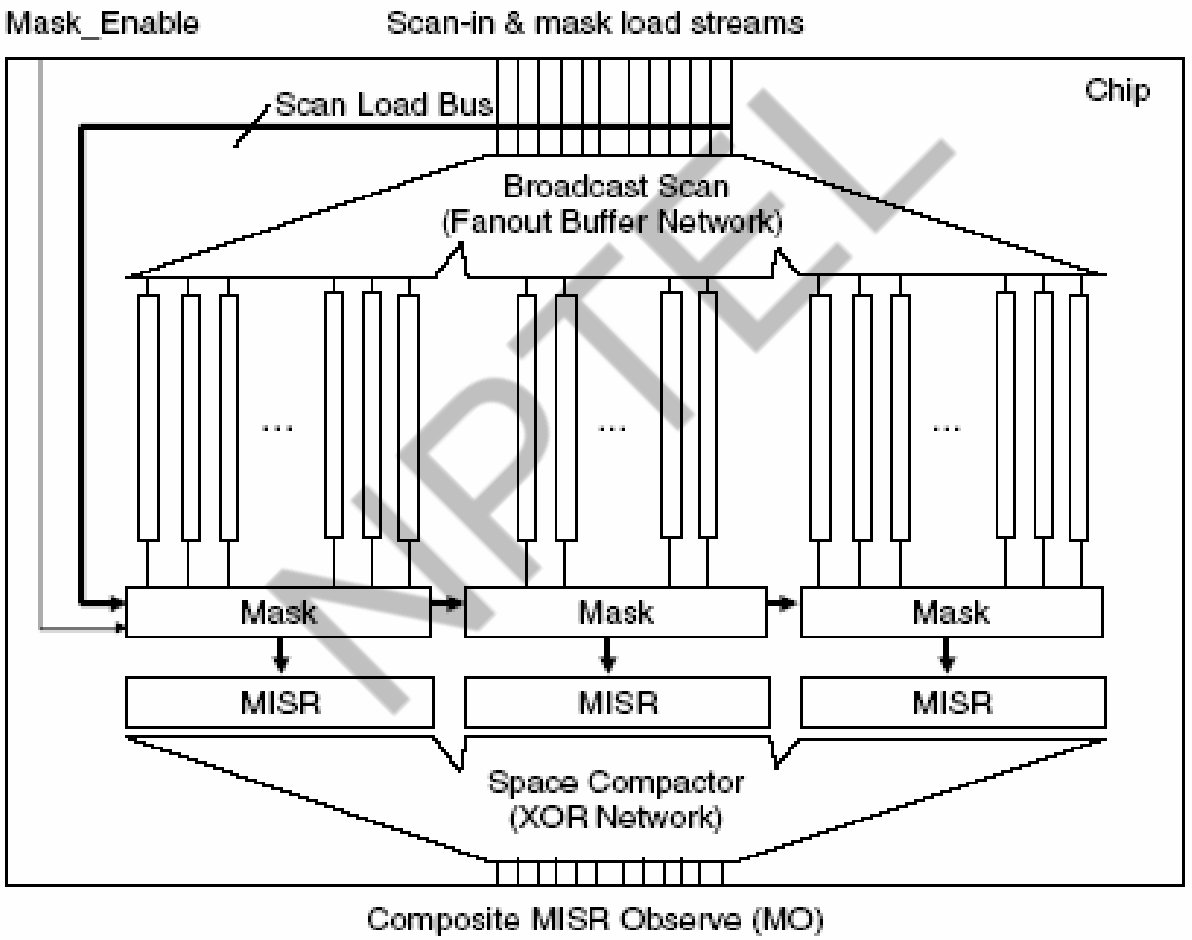
Industry solutions categories

- Linear-decompression-based schemes
  - Two steps
    - ETCompression, LogicVision
    - TestKompress, Mentor Graphics
    - SOCBIST, Synopsys
- Broadcast-scan-based schemes
  - Single step
    - SPMISR+, Cadence
    - VirtualScan and UltraScan, SynTest
    - DFT MAX, Synopsys

Industry practices

- OPMISR+
  - Cadence
  - Roots in IBM ‘s logic BIST and ATPG technology

General scan architecture for OPMISR+



Concluding remarks

- Test compression is
  - An effective method for reducing test data volume and test application time with relatively small cost
  - An effective test structure for embedded hard cores
  - Easy to implement and capable of producing high-quality tests
  - Successfully as part of design flow
- Need to unify different compression architectures