Chapter 6

Test Compression

What is this chapter about?

- Introduce the basic concepts of test data compression
- Focus on stimulus compression and response compaction techniques
- Present and discuss commercial tools on test compression

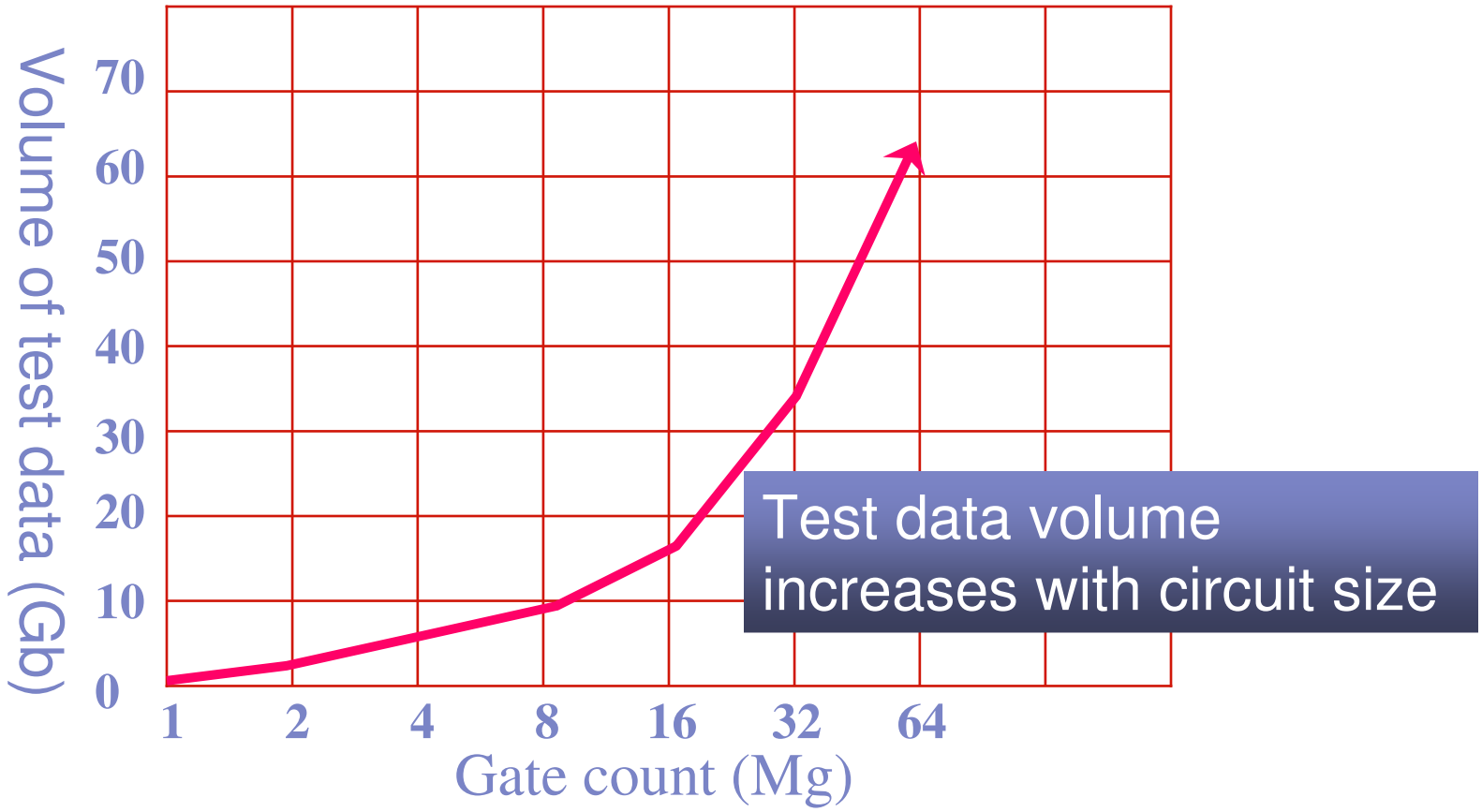
Test Compression

- 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

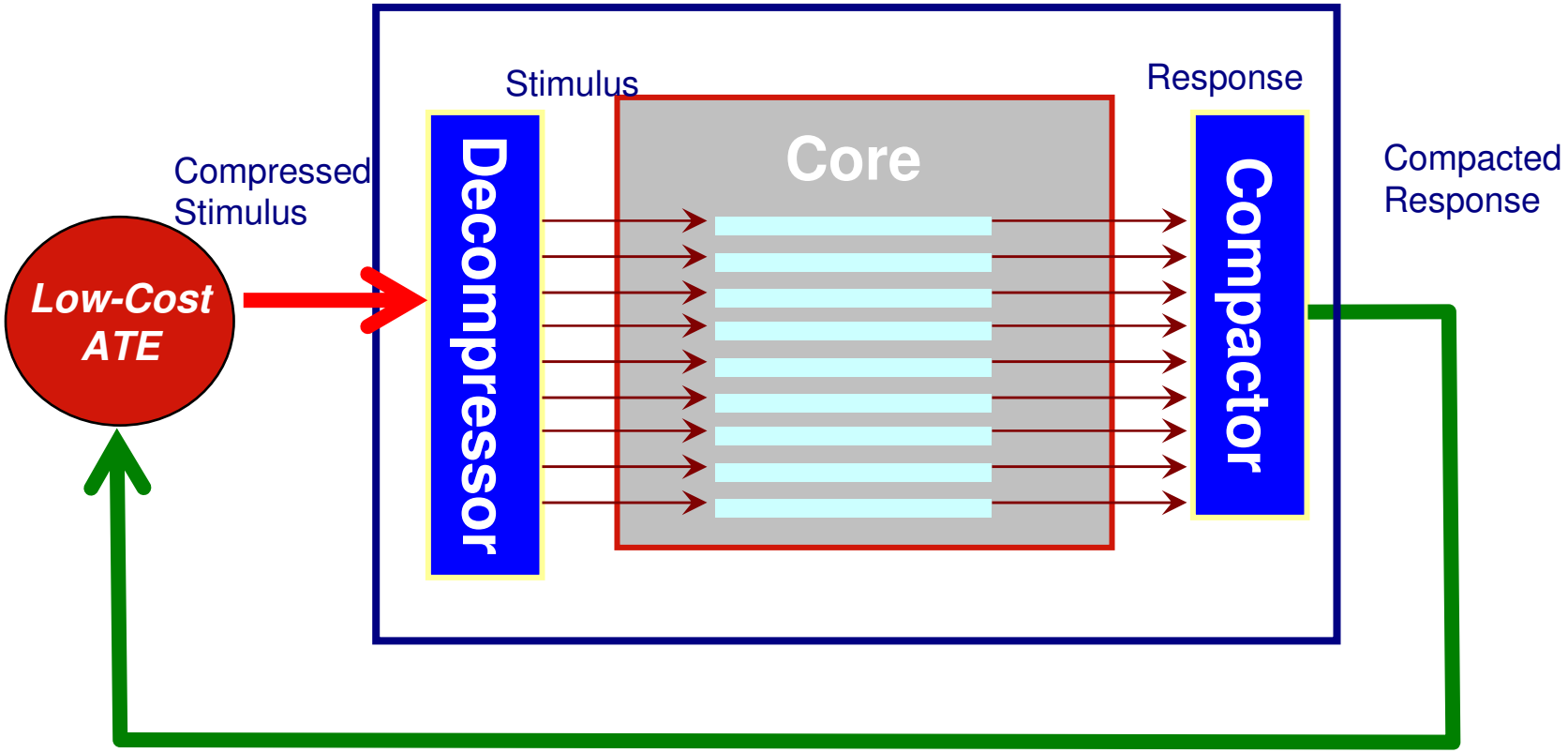


(Source: Blyler, Wireless System Design, 2001)

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

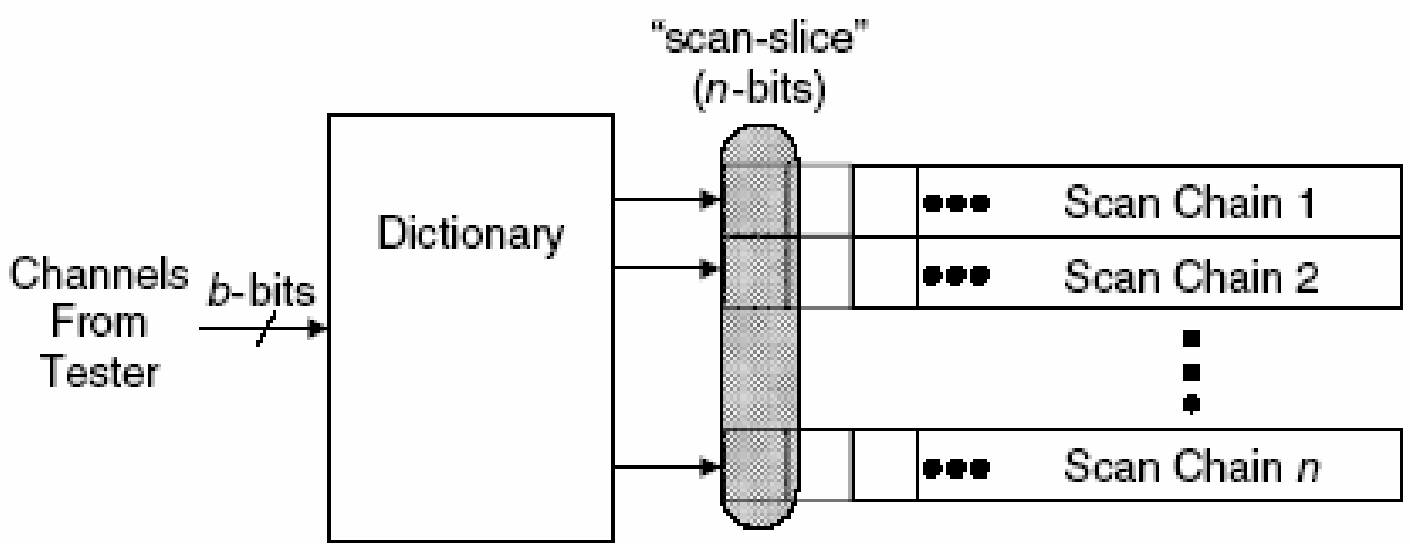
- Code-based schemes
- Linear-decompression-based schemes
- Broadcast-scan-based schemes

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)



Code-based schemes

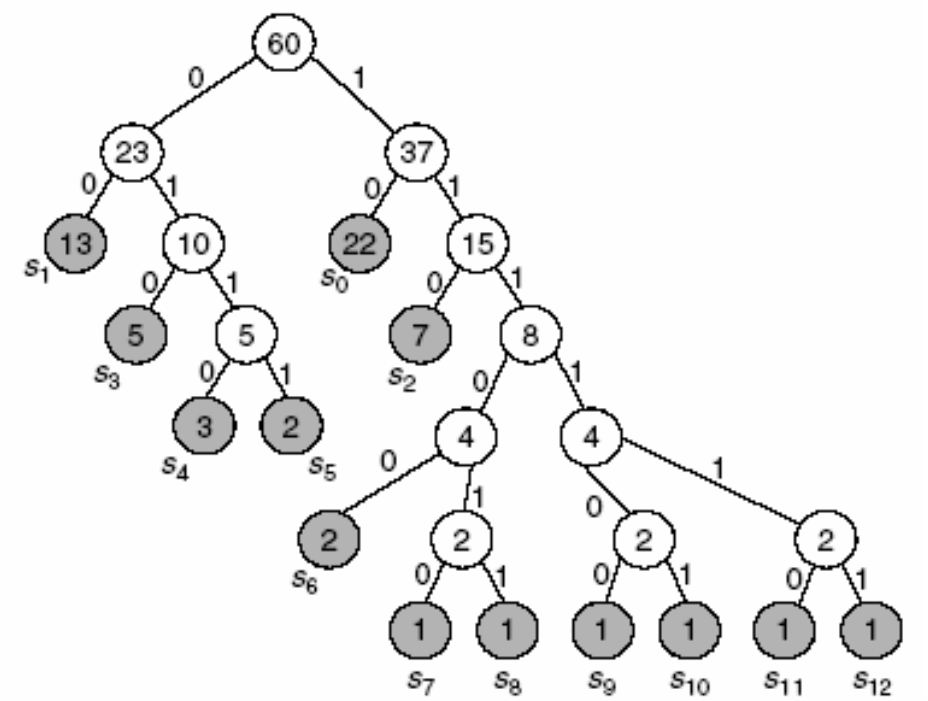
- Huffman code (fixed-to-variable)

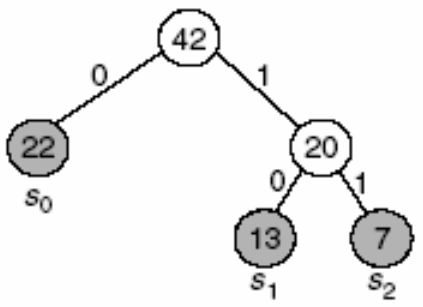
Symbol	Frequency	Pattern	Huffman Code	Selective Code
S_0	22	0010	10	10
S_1	13	0100	00	110
S_2	7	0110	110	111
S_3	5	0111	010	00111
S_4	3	0000	0110	00000
S_5	2	1000	0111	01000
S_6	2	0101	11100	00101
S_7	1	1011	111010	01011
S_8	1	1100	111011	01100
S_9	1	0001	111100	00001
S_{10}	1	1101	111101	01101
S_{11}	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)

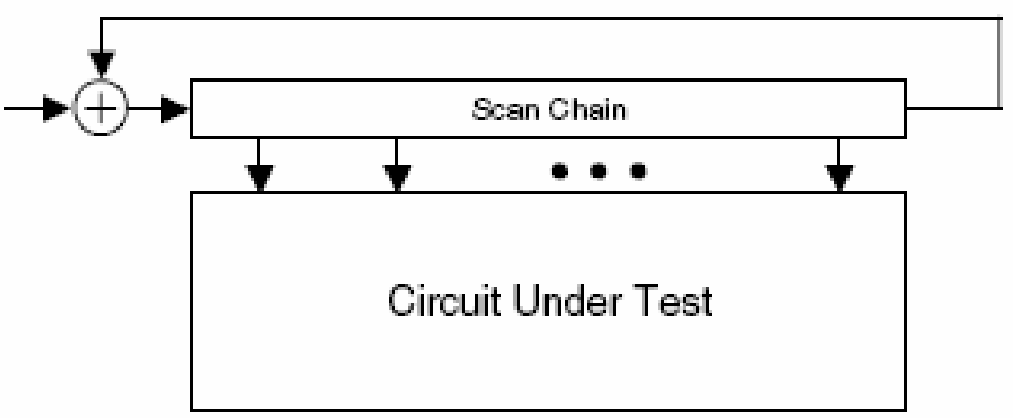




Code-based schemes

- 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)

T_D = 001 00001 0001 00001 00001 0000 01 001 00000001 00 01

$l_1 = 2$ $l_2 = 4$ $l_3 = 3$ $l_4 = 4$ $l_5 = 4$ $l_6 = 5$ $l_7 = 2$ $l_8 = 7$ $l_9 = 3$

Using Golomb code shown in Table 6.4

T_E = 010 1000 011 1000 1000 1001 010 1011 011

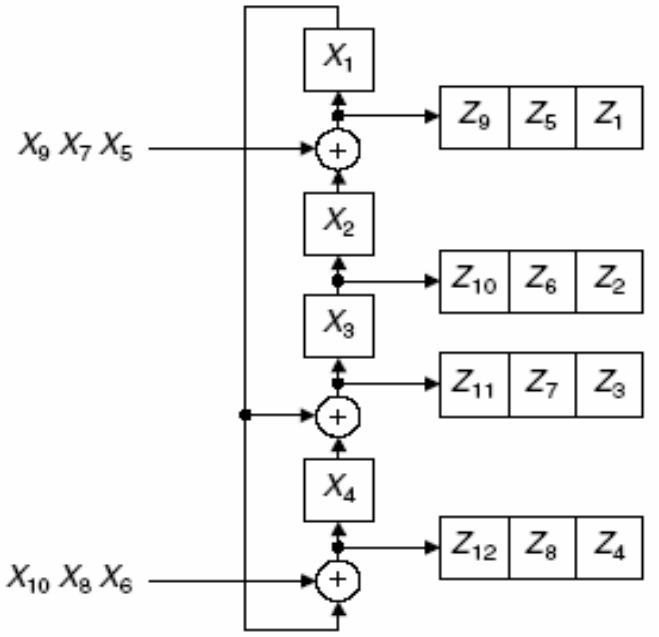
The length of T_D is 43 bits

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



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

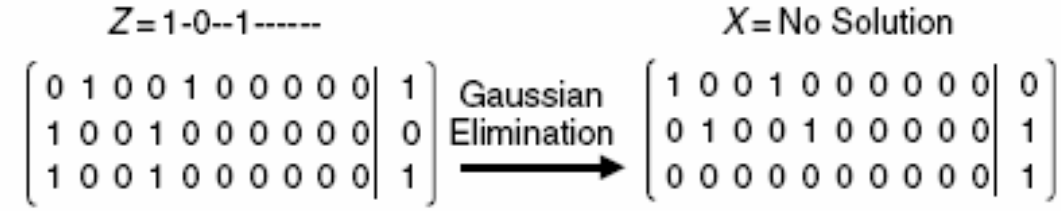
$$Z_5 = X_3 \oplus X_7$$
$$Z_6 = X_1 \oplus X_4$$
$$Z_7 = X_1 \oplus X_2 \oplus X_5 \oplus X_6$$
$$Z_8 = X_2 \oplus X_5 \oplus X_8$$

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

Linear-decompression-based schemes

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

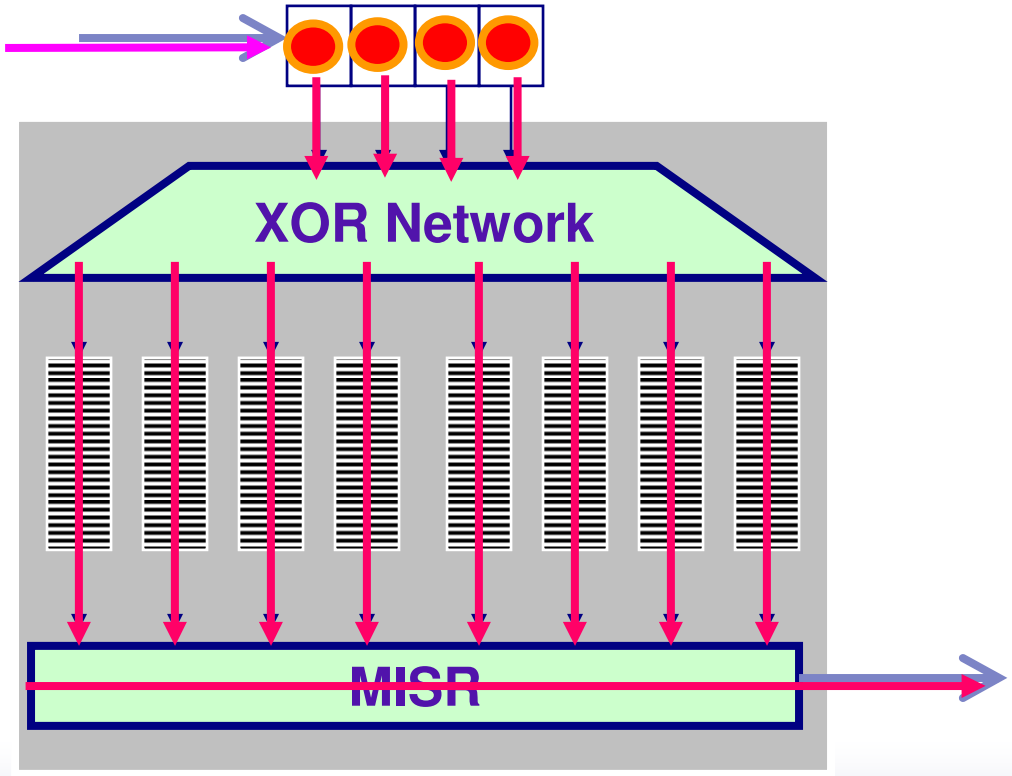
Linear-decompression-based schemes



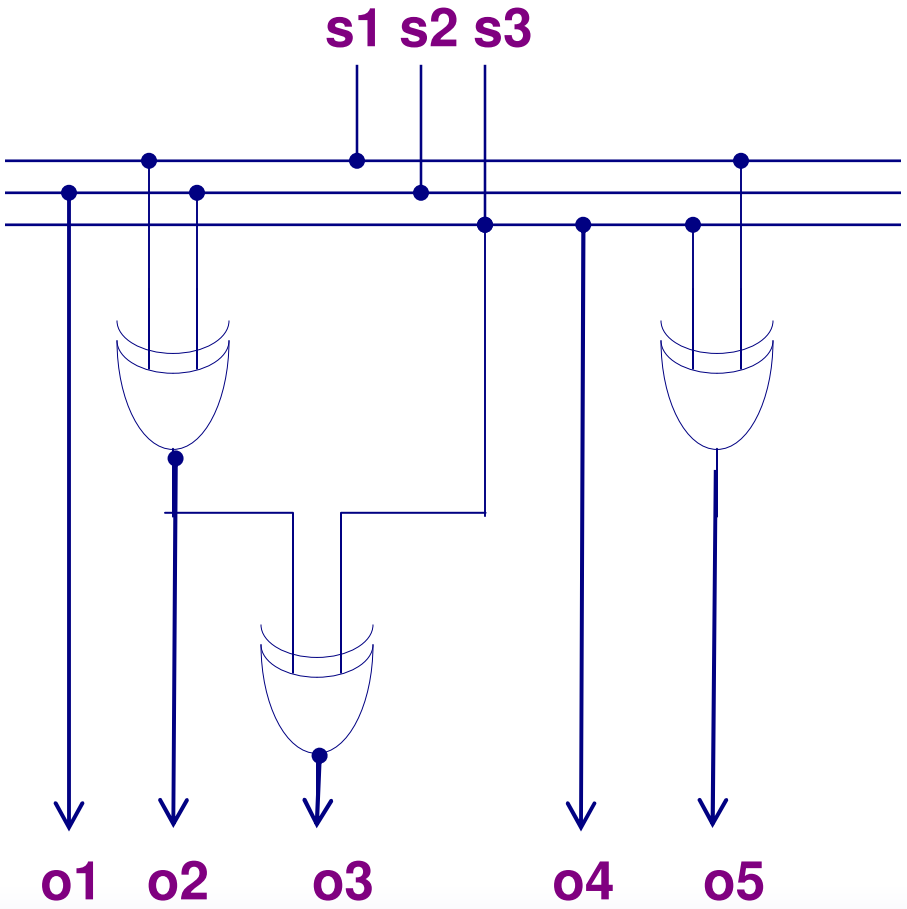
Linear-decompression-based schemes

- Combinational linear decompressors

XOR Network

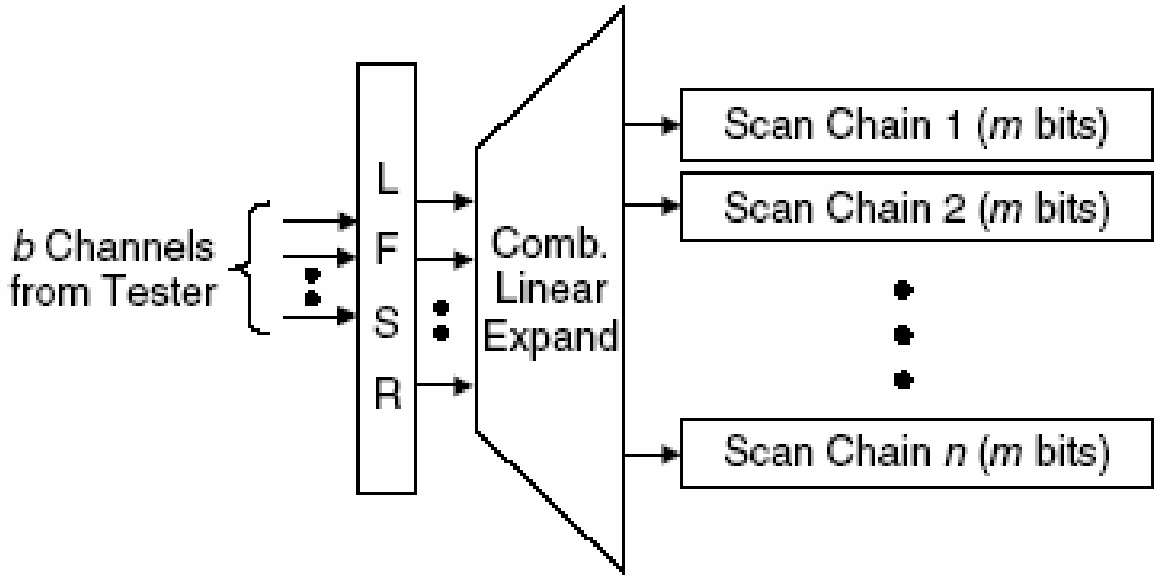


XOR network: a 3-to-5 example



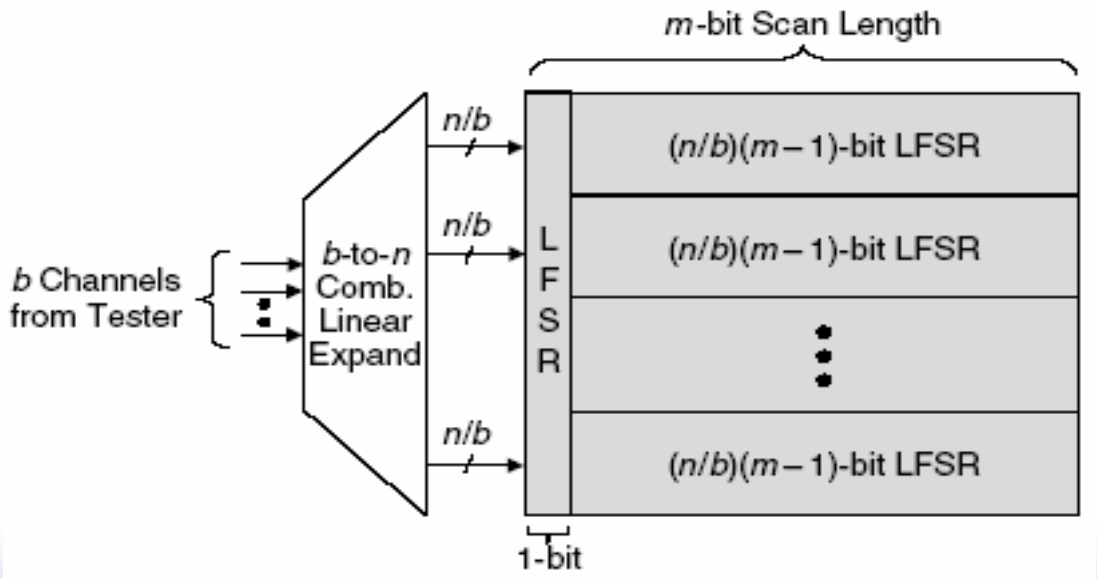
Linear-decompression-based schemes

- Fixed-length sequential linear decompressors



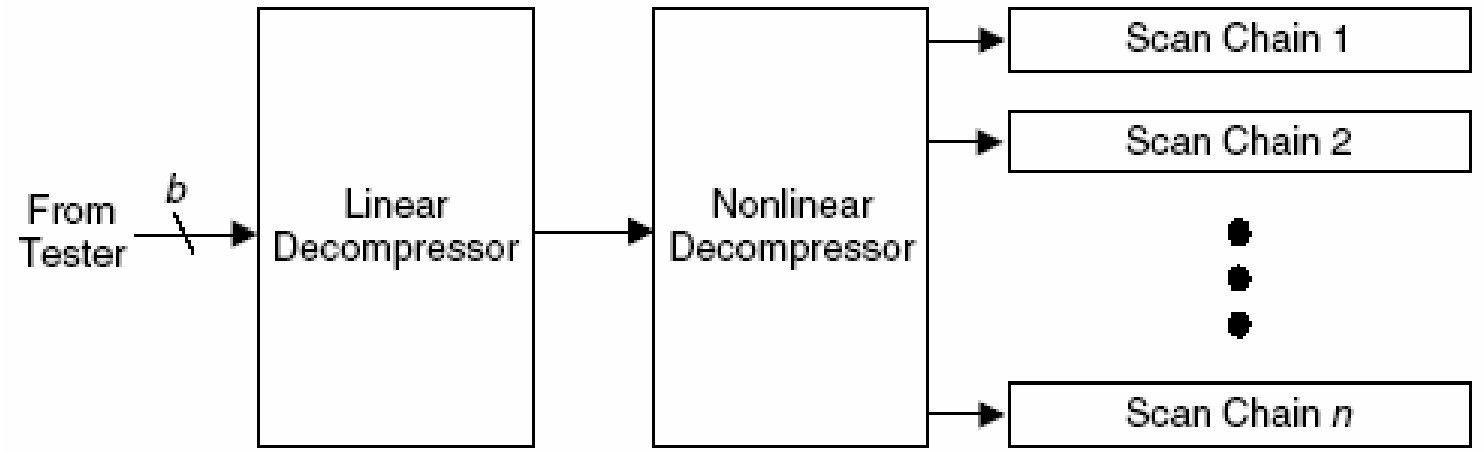
Linear-decompression-based schemes

- 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

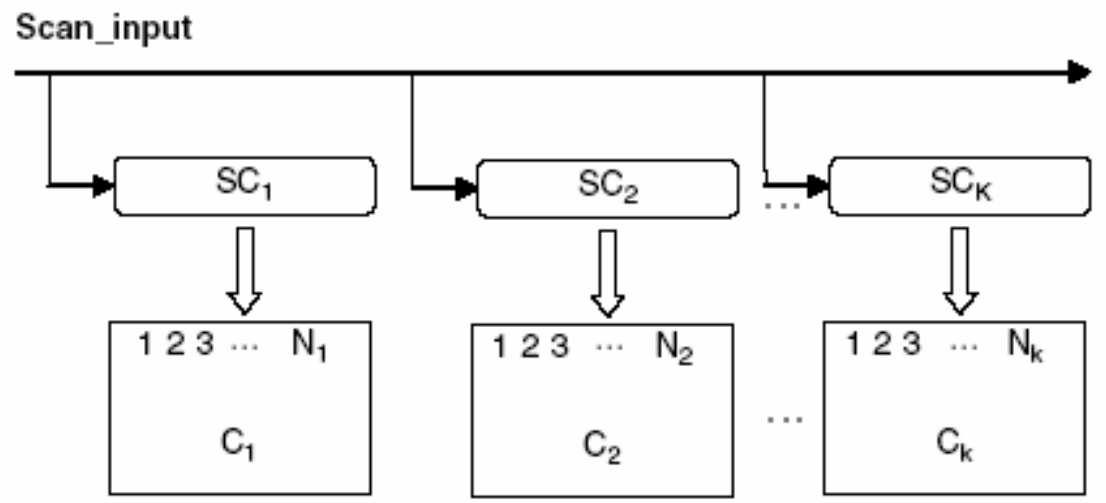


Test stimulus compression

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

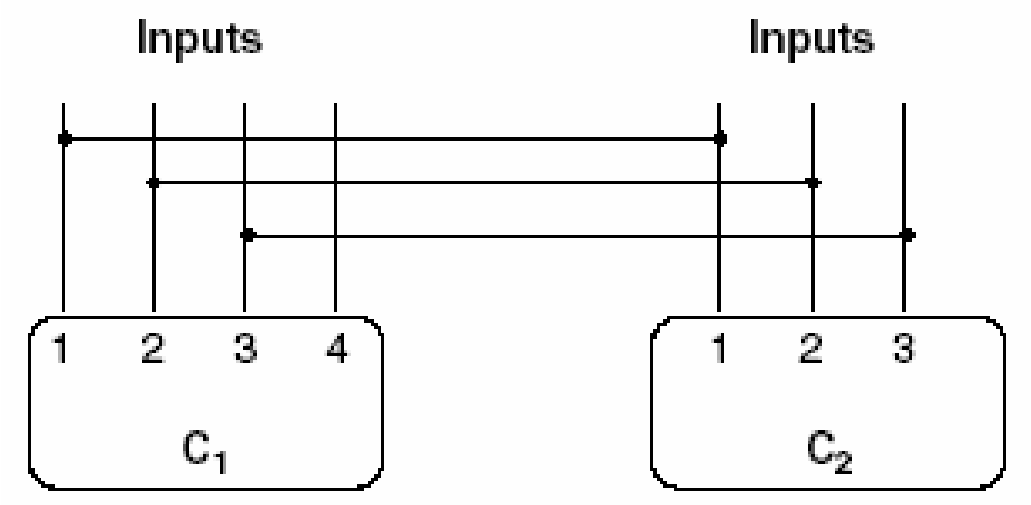
Broadcast-scan-based schemes

- Broadcast scan



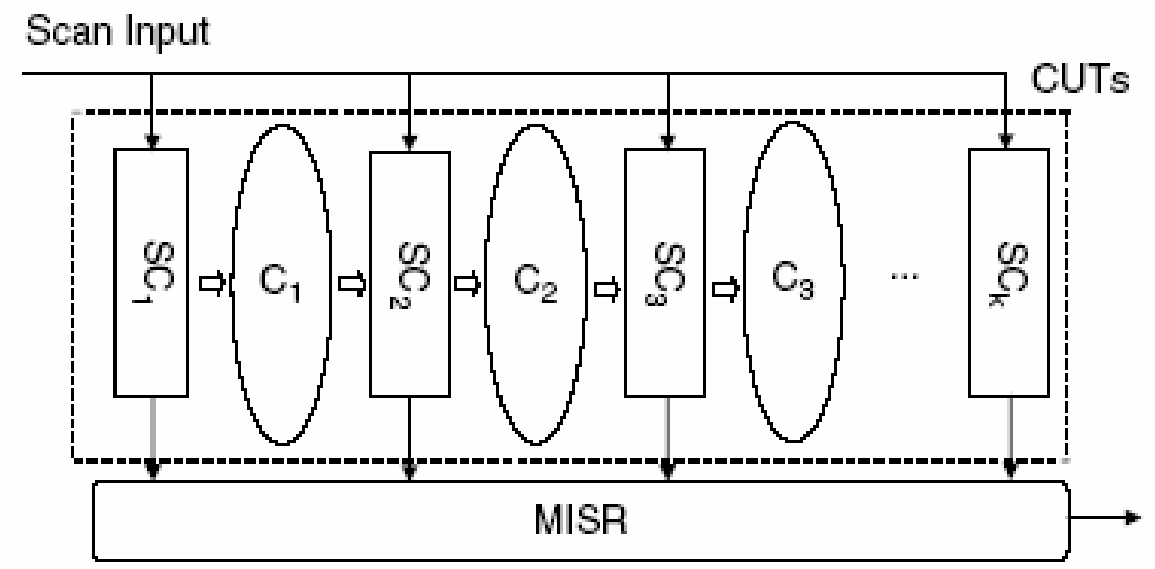
Generate patterns for broadcast scan

- Force ATPG tool to generate patterns for broadcast scan



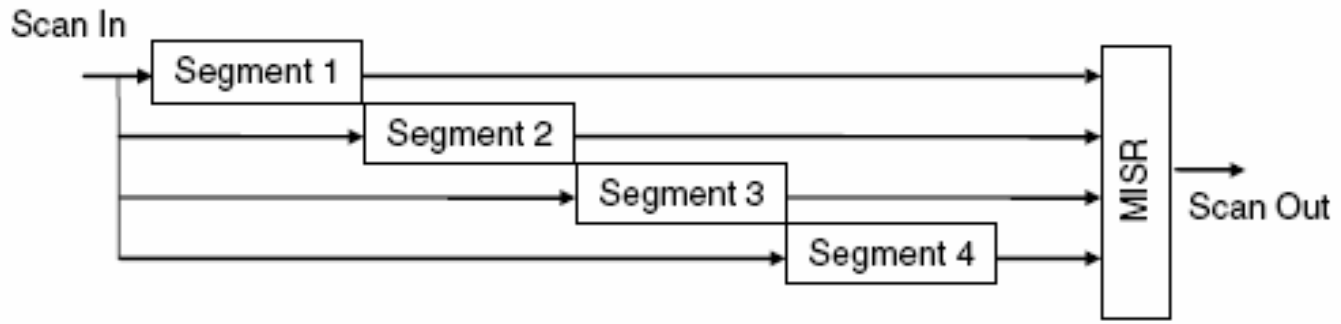
Broadcast scan for a pipelined circuit

- Broadcast scan for a pipelined circuit

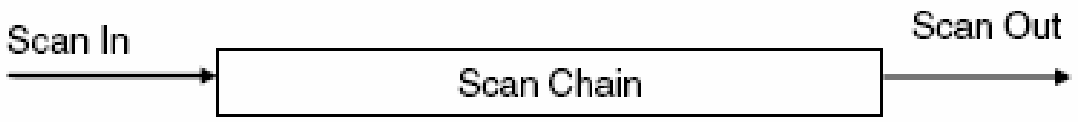


Broadcast-scan-based schemes

- 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

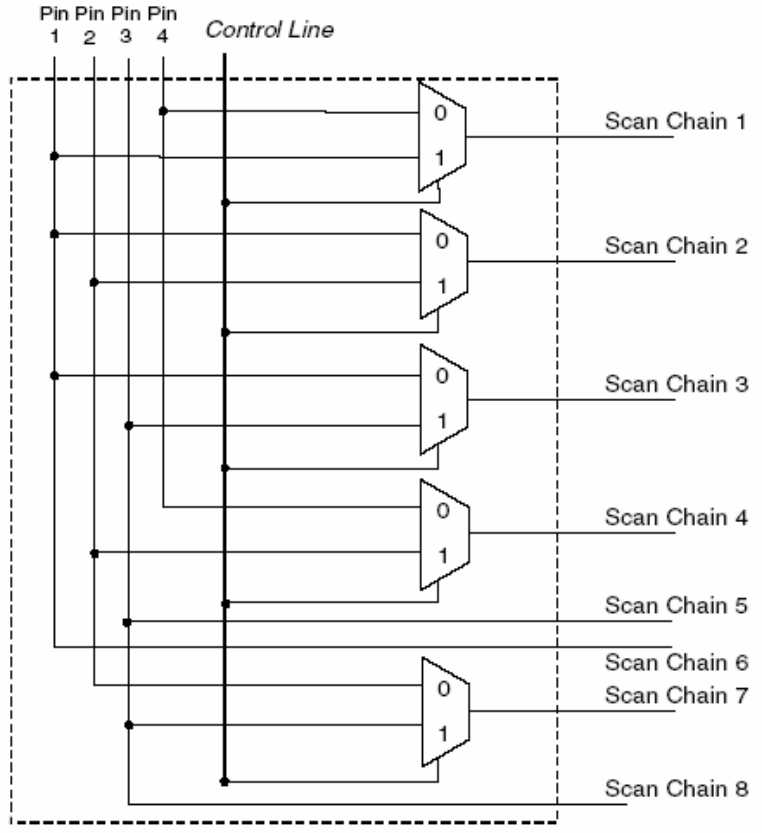
Broadcast-scan-based schemes

- 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



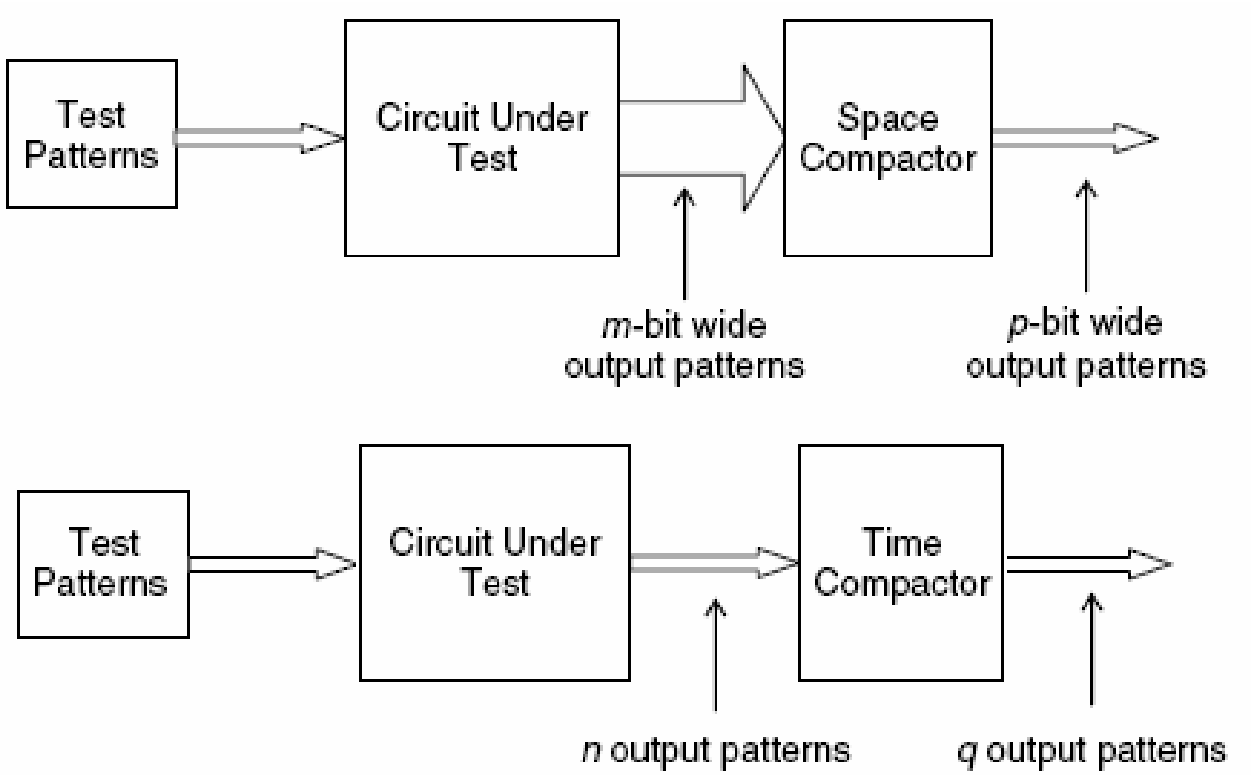
Broadcast-scan-based schemes

- 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

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

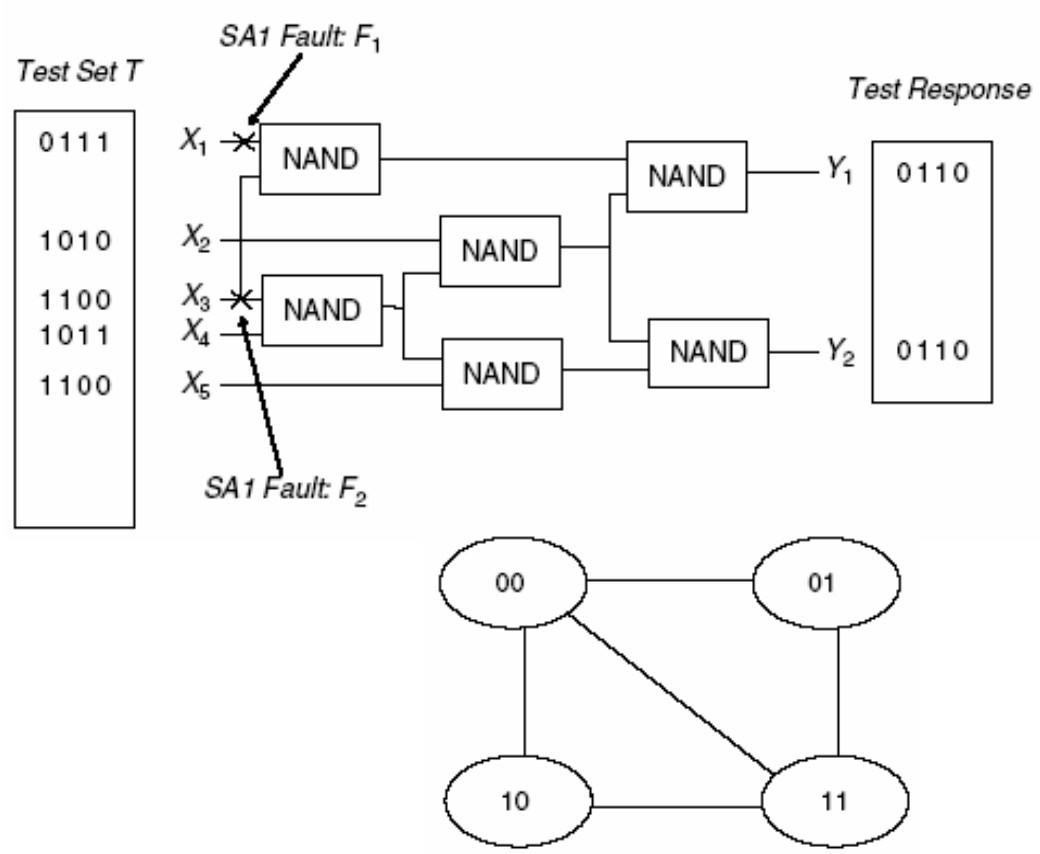
Theorem 6.1

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

- X-compact

Theorem 6.3

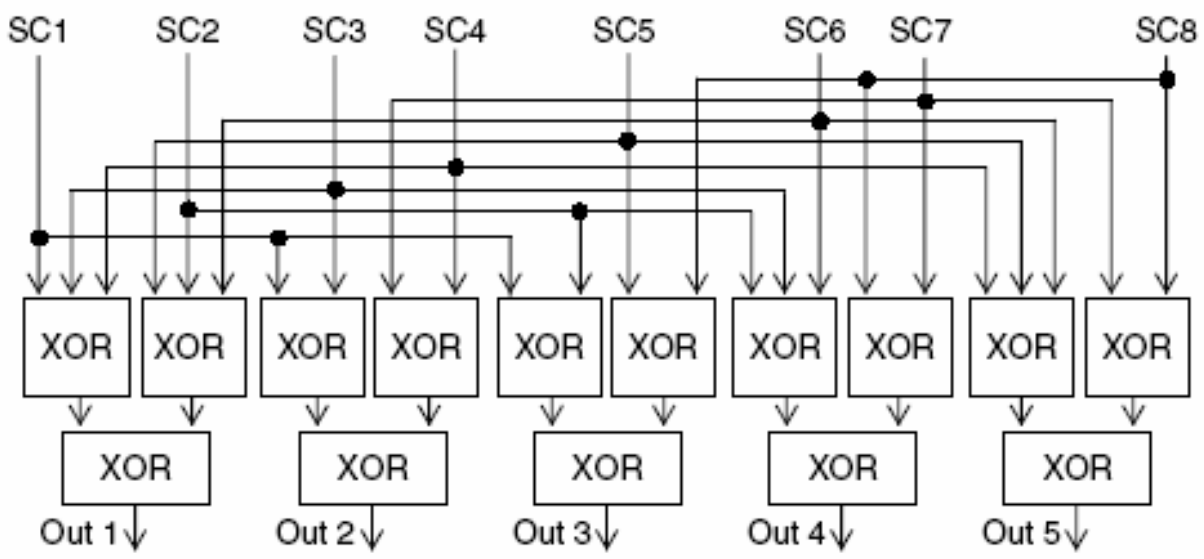
If only a single scan chain produces an error at any scan-out cycle, the X-compactor is guaranteed to produce errors at the X-compactor outputs at that scan-out cycle, if and only if no row of the X-compact matrix contains all 0's.

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

$$S=\begin{bmatrix} SC1 \\ SC2 \\ SC3 \\ SC4 \\ SC5 \\ SC6 \\ SC7 \\ SC8 \end{bmatrix} \qquad M=\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix} \qquad O=\begin{bmatrix} O1 \\ O2 \\ O3 \\ O4 \\ O5 \end{bmatrix}$$

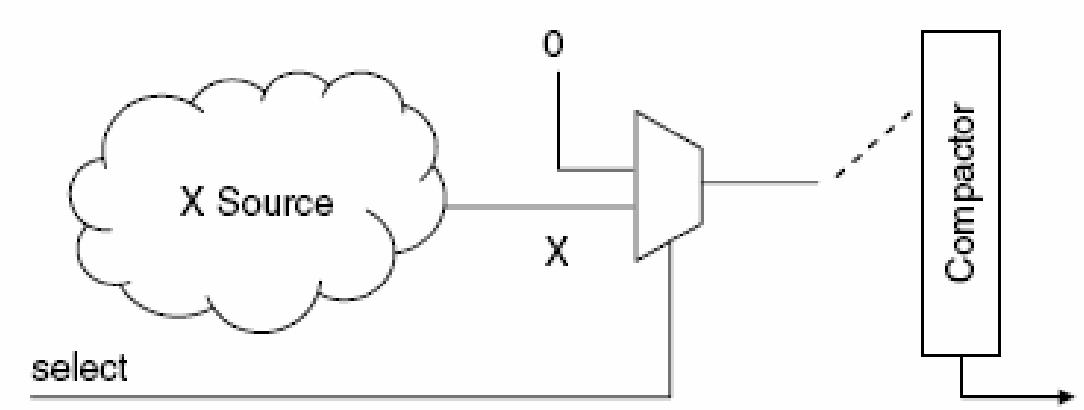
$$\mathbf{M^T \; X \; S \; = \; O}$$

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

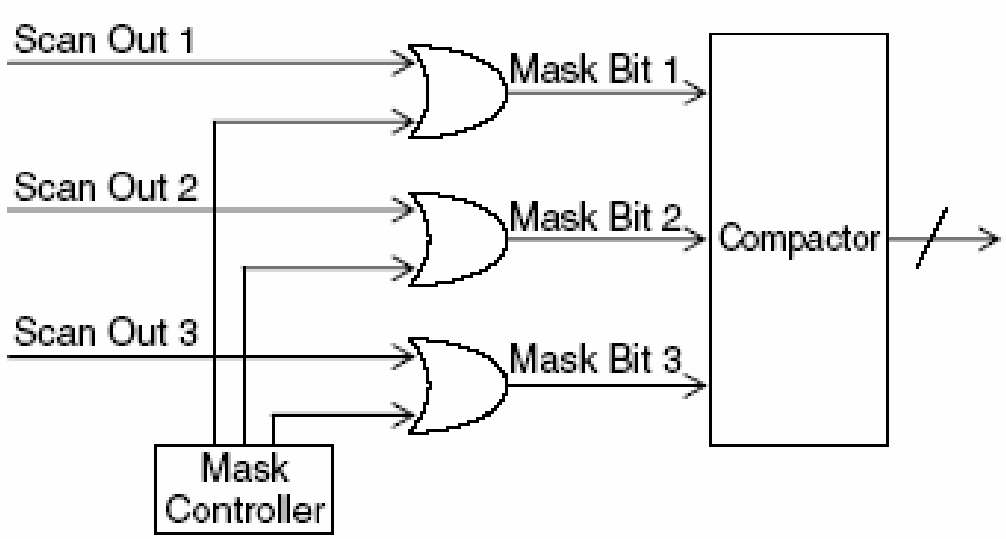


Space compaction

- 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

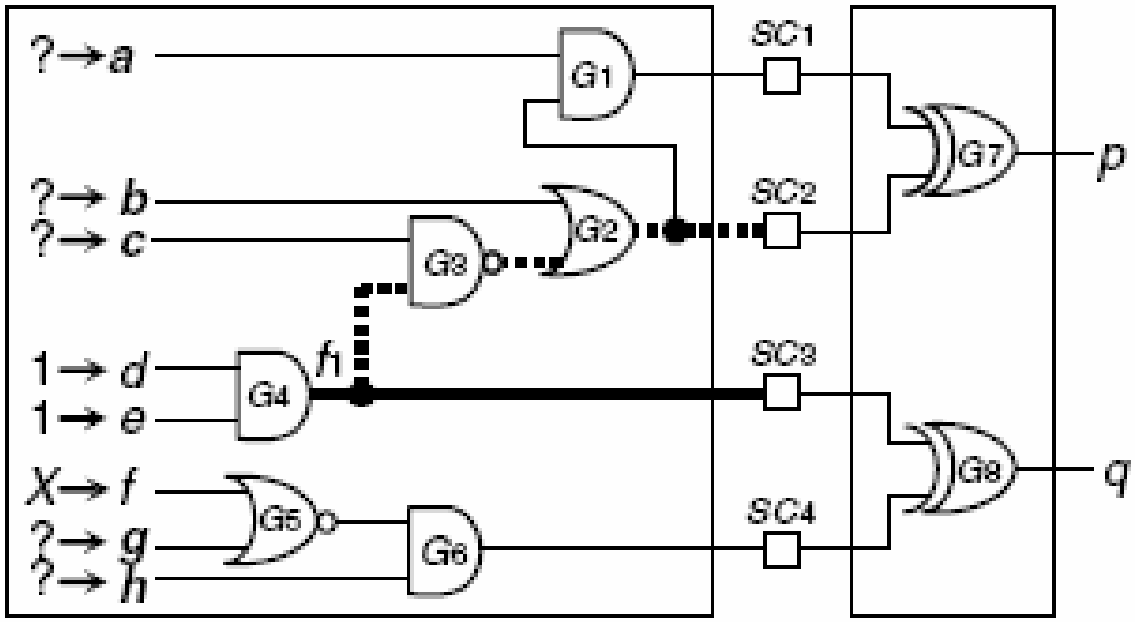


Space compaction

- X-impact
  - Simply use ATPG to algorithmically handle the impact of residual x's on the space compactor
  - Without adding any extra circuitry

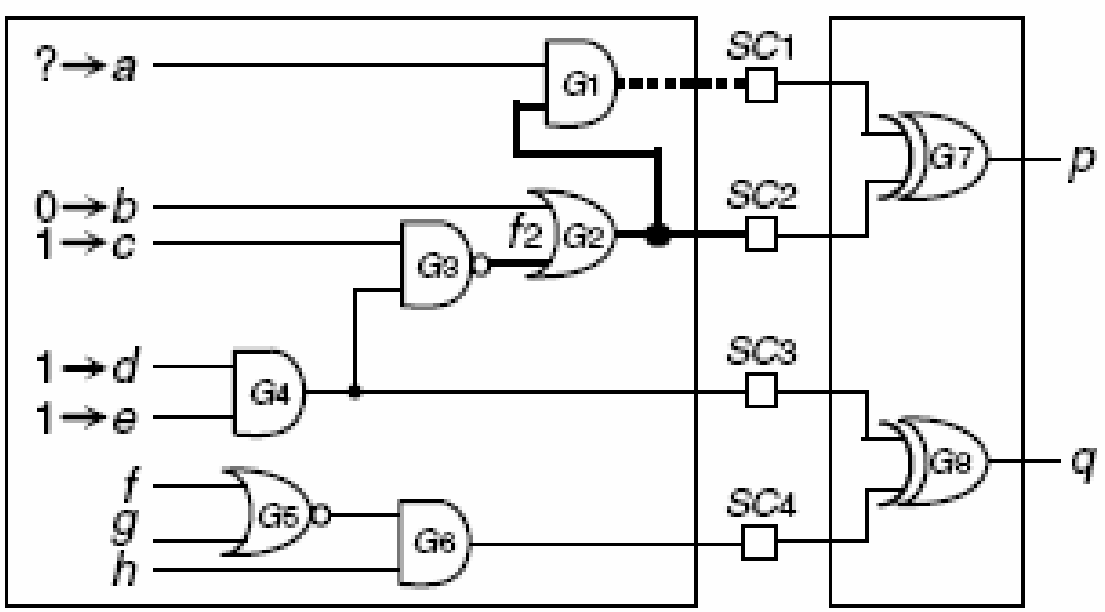
Space compaction

- Handling of X-impact



Space compaction

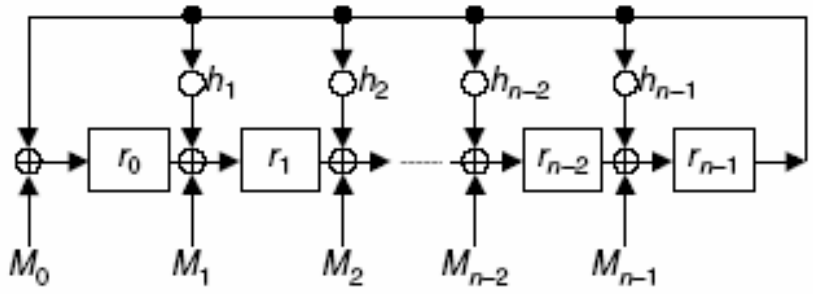
- Handling of aliasing



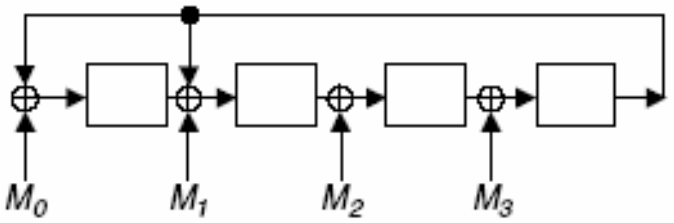
Test response compaction

- 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)



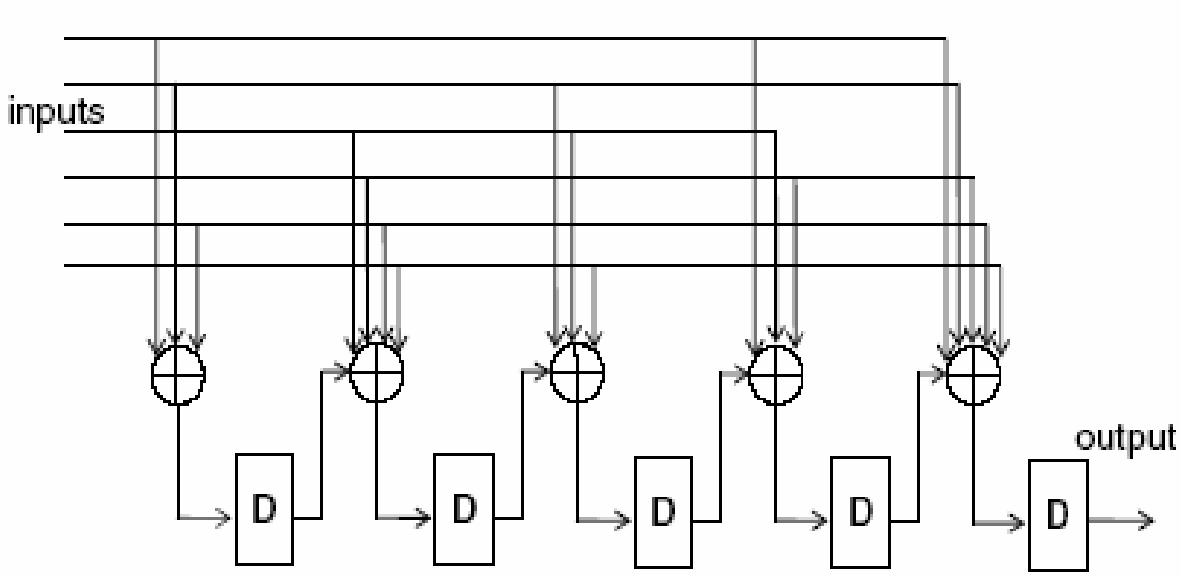
$$f(x) = 1 + h_1 x + h_2 x^2 + \cdots + h_{n-1} x^{n-1} + x^n$$



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

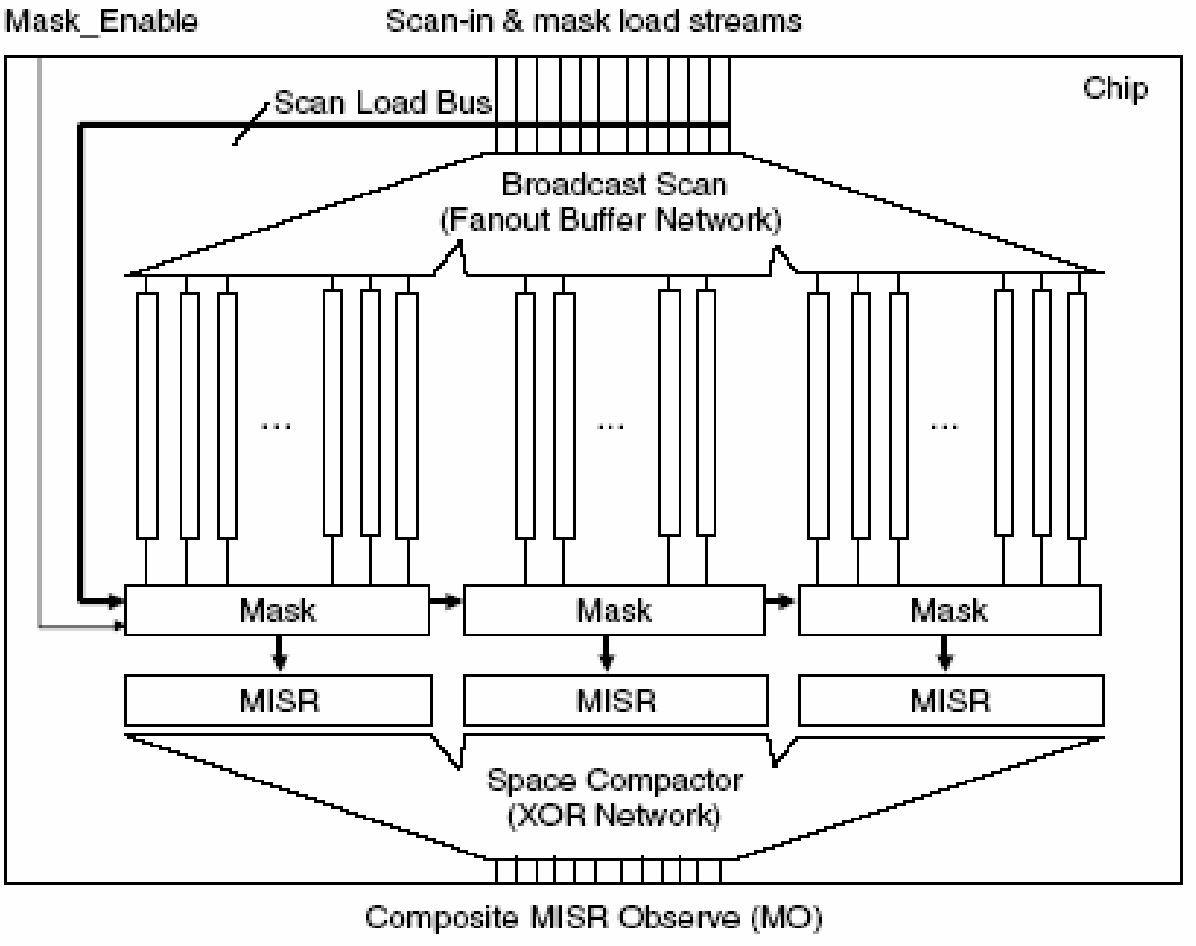
Industry solutions categories

- 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

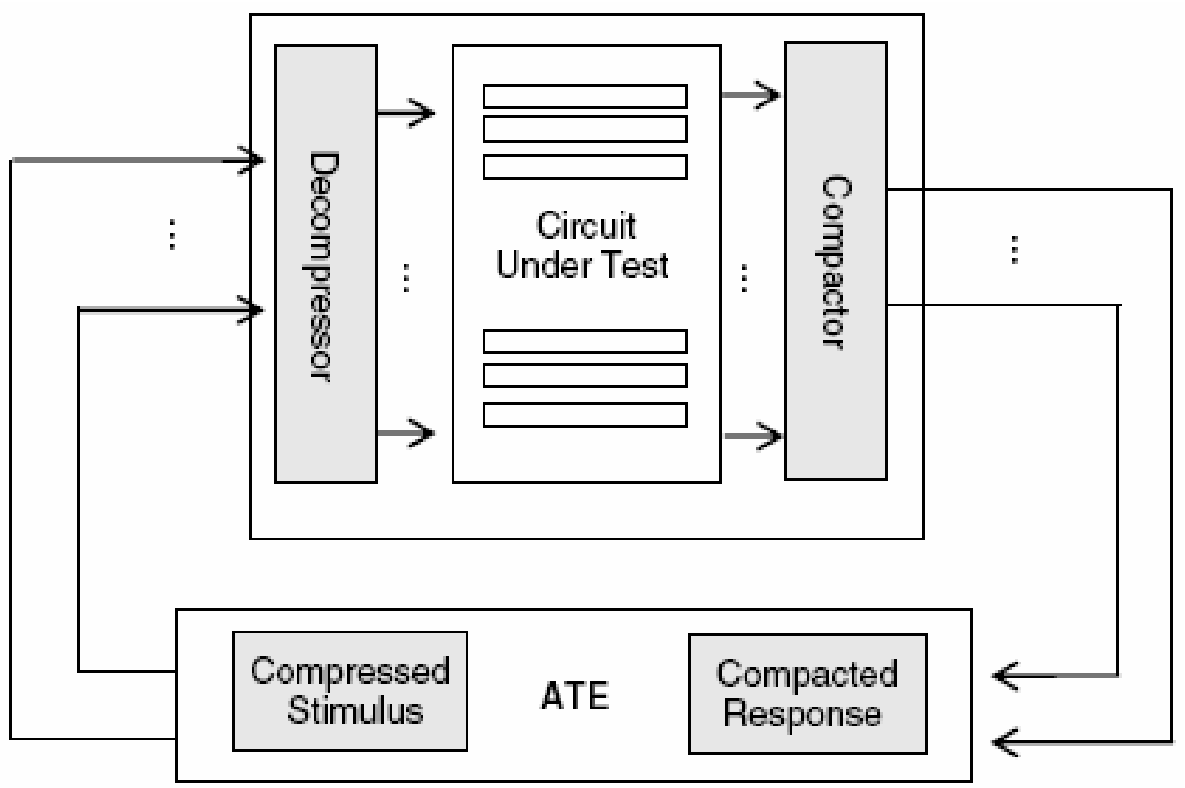
General scan architecture for OPMISR+



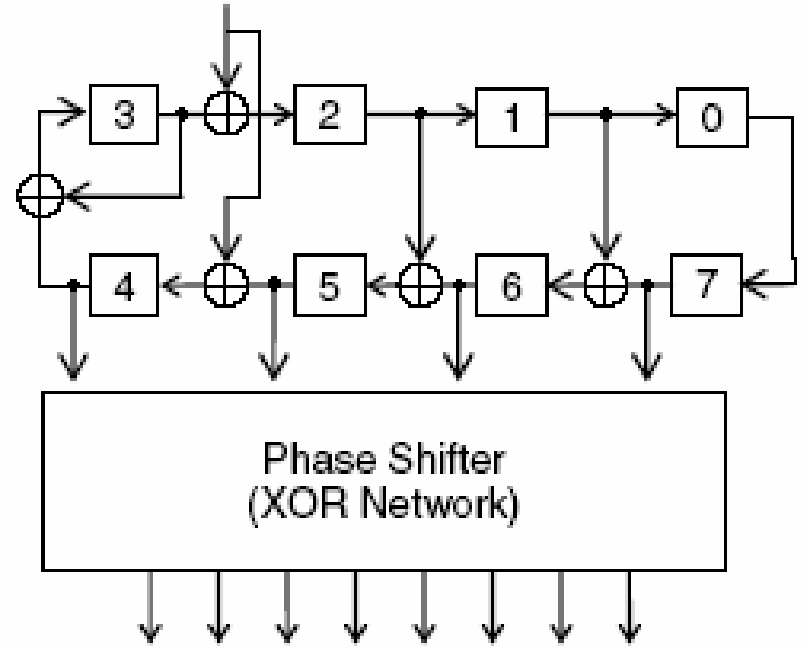
Industry practices

- Embedded Deterministic Test (TestKompress)
  - Mentor Graphics
  - First commercially available on-chip test compression product

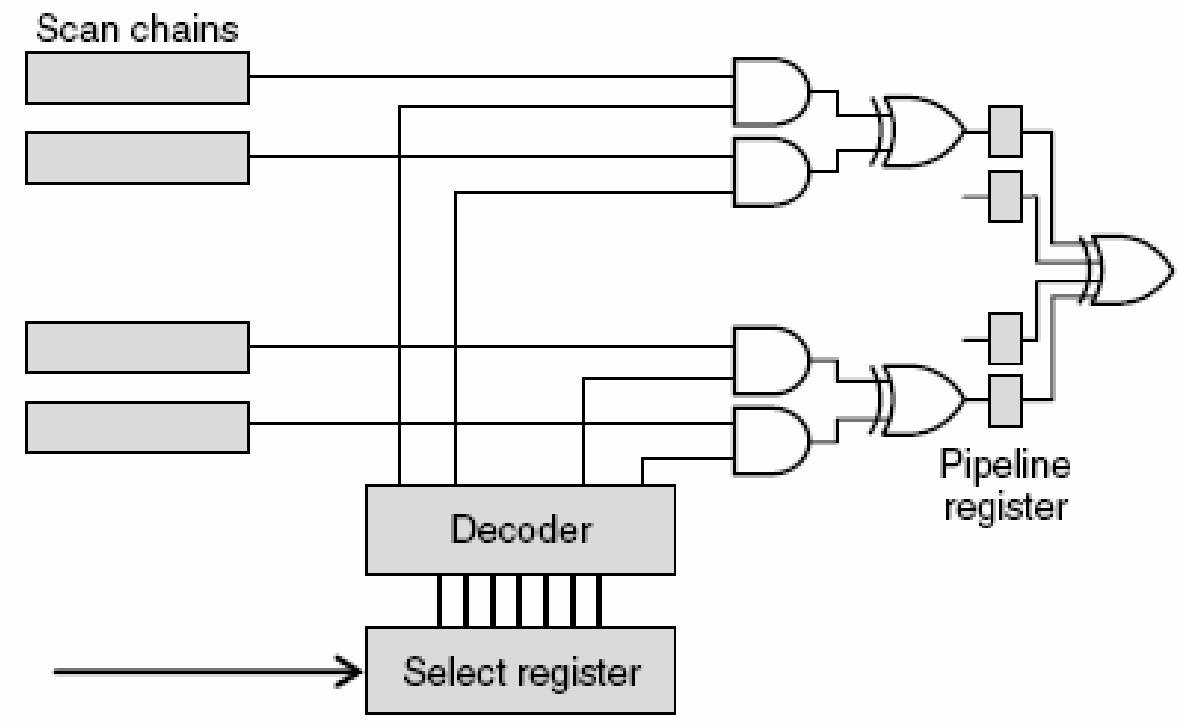
EDT (TestKompression) architecture



TestKompress stimuli compression



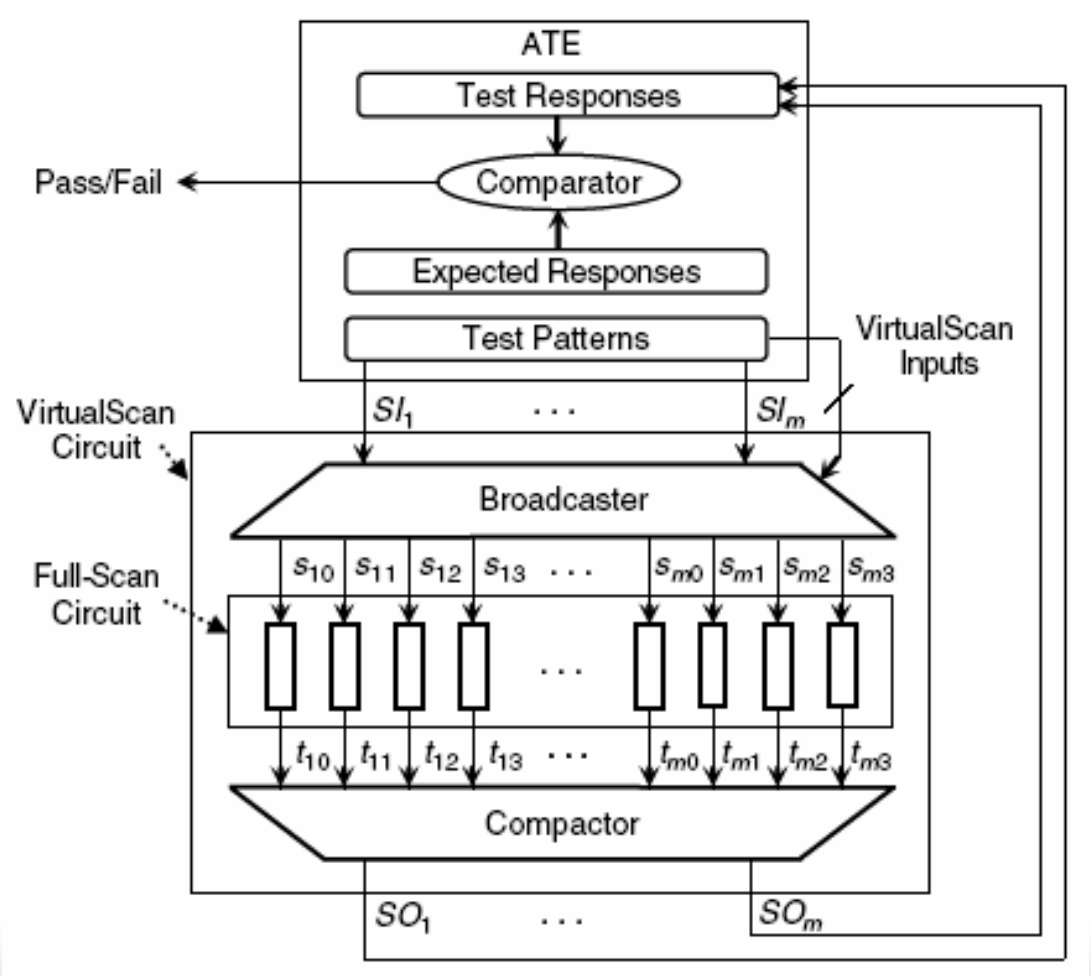
TestKompress response compaction



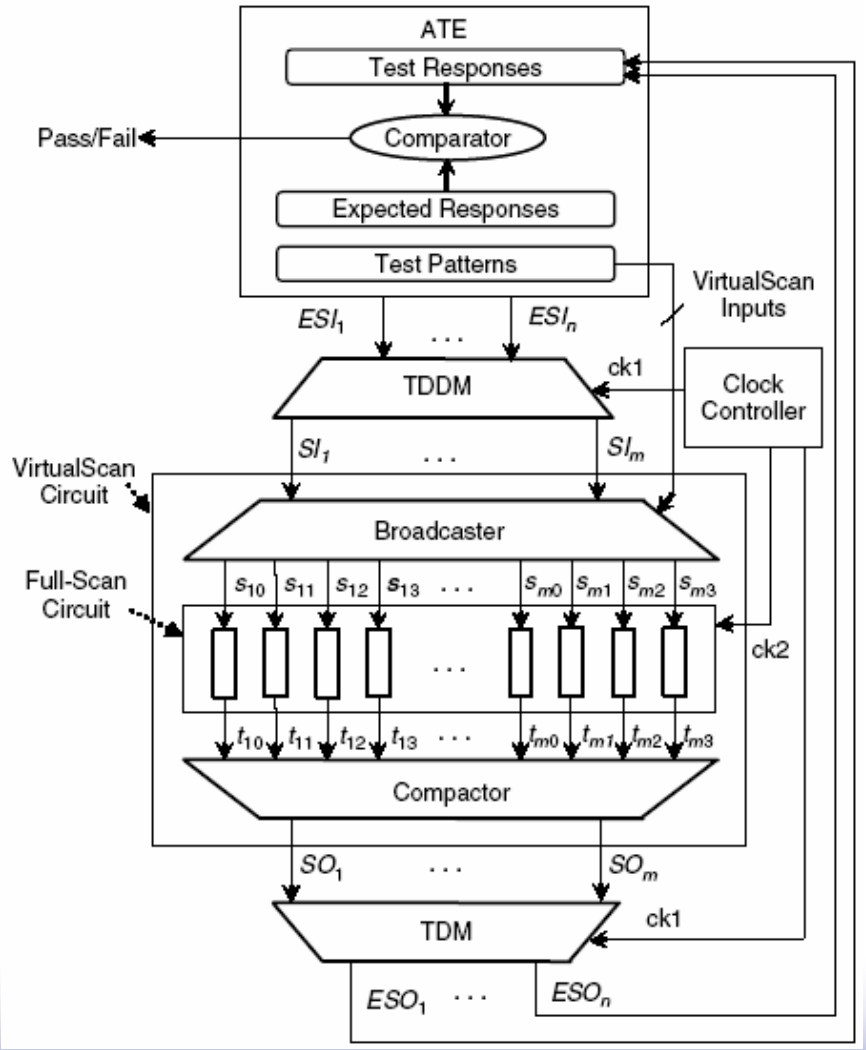
Industry practices

- Virtual Scan and UltraScan
  - SynTest
  - First commercial product based on the broadcast scan scheme using combinational logic for pattern decompression

VirtualScan architecture



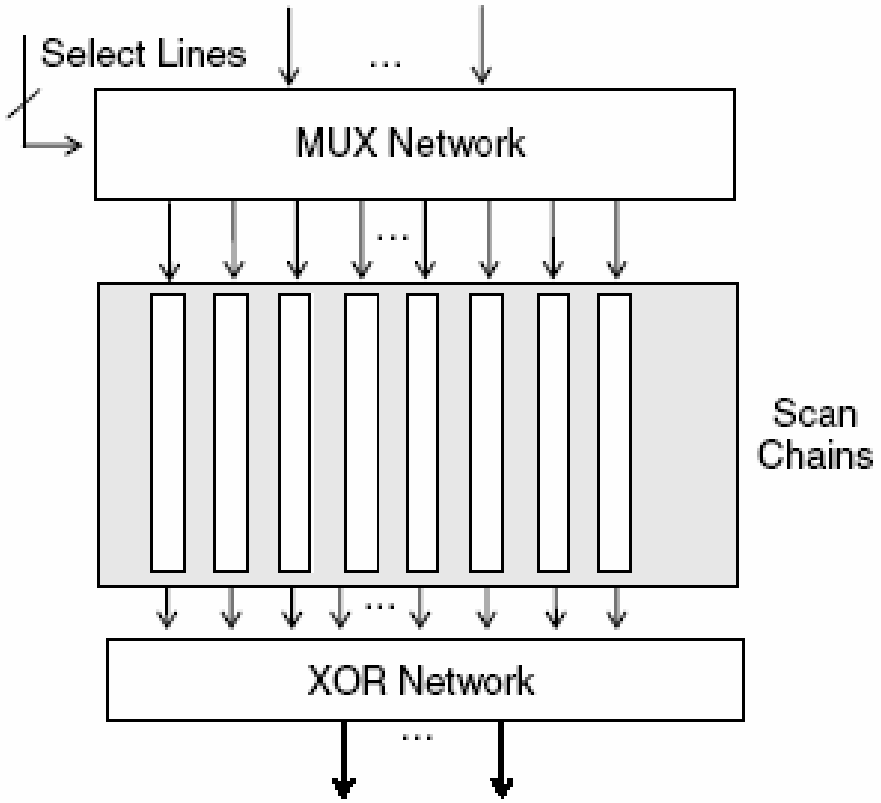
UltraScan architecture



Industry practices

- Adaptive Scan
  - Synopsys
  - Designed to be the next generation scan architecture

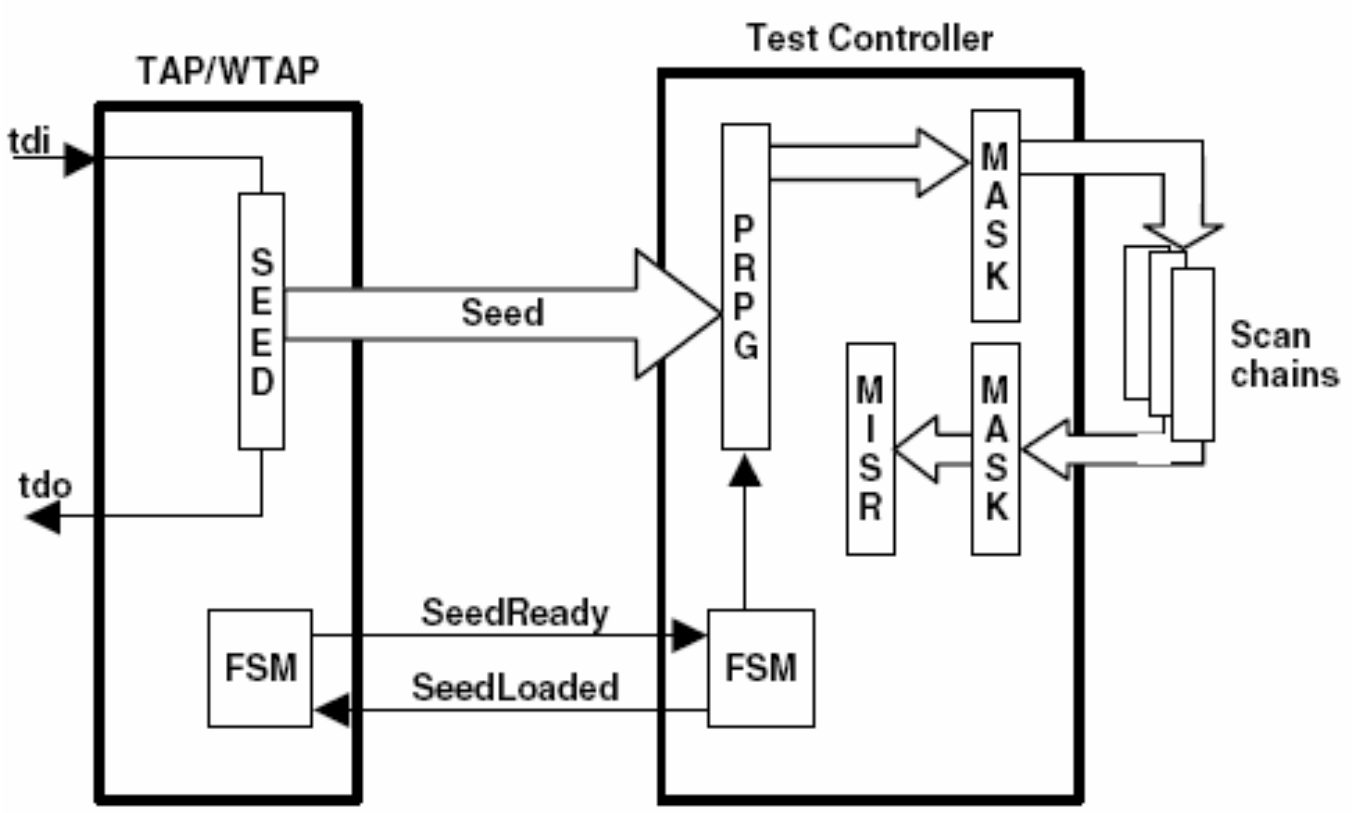
Adaptive scan architecture



Industry practices

- ETCompression
  - LogicVision
  - Built upon *embedded logic test (ELT)* technology

ETCompression architecture



Summary of industry practices

Industry Practice	Stimulus Decompressor	Response Compactor
OPMISR+	Broadcast scan (Illinois scan)	MISR with XOR network
TestKompress	Ring generator	XOR network
VirtualScan	Combinational logic network	XOR network
DFT MAX	Combinational MUX network	XOR network
ETCompression	(Reseeding) PRPG	MISR
UltraScan	TDDM	TDM

MISR: multiple-input signature register

MUX: multiplexers

PRPG: pseudo-random pattern generator

TDDM: time-division demultiplexer

TDM: time-division multiplexers

XOR: exclusive-OR

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