Chapter 8

Test Compression

Acknowledgements:
Mainly based on the lecture notes of
Chapter 6, "VLSI Test Principles and Architectures"

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**

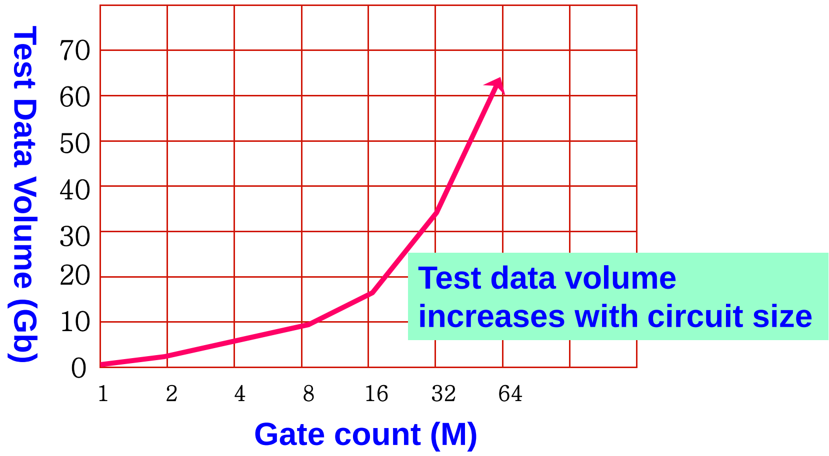
Test Compression

- 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?**
  - Test vectors have a lot of “don’t care” (X’s)

Test Data Volume v.s. Gate Count

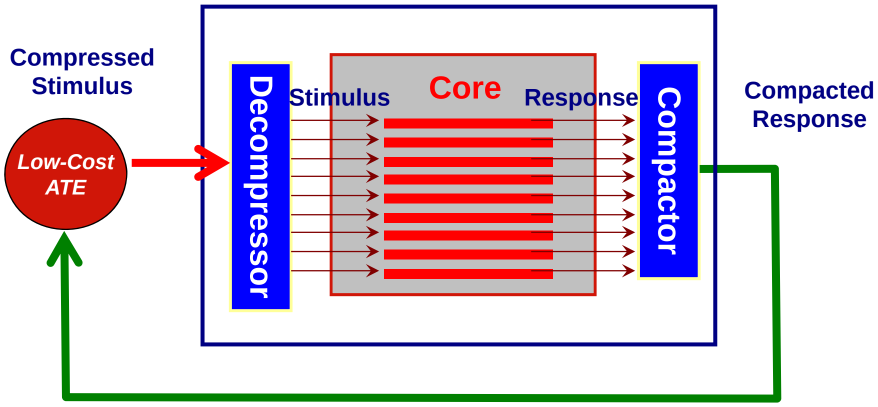


(Source: Blyler, Wireless System Design, 2001)

Test Compression Categories

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

Architecture for Test Compression



ch8-7

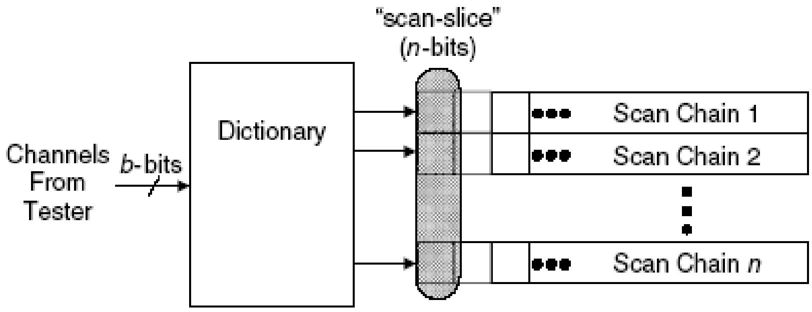
Test Stimulus Compression

- ❑ Code-based schemes
  - Dictionary code (fixed-to-fixed)
  - Huffman code (fixed-to-variable)
  - Run-length code (variable-to-fixed)
  - Golomb code (variable-to-variable)
- ❑ Linear-decompression-based schemes
- ❑ Broadcast-scan-based schemes

ch8-8

Dictionary Code

- Dictionary code (fixed-to-fixed)



A test vector is considered as a two-dimensional image
In multiple scan chains (e.g., n scan chains as shown)

Huffman Code

- 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 01
0010 0100 0010 0110 0000 0110 0010 0100 0110 0010 00
0010 0110 0010 0010 0010 0100 0100 0110 0010 0010 1
0001 0100 0010 0111 0010 0010 0111 0111 0100 0100 1
1100 0100 0100 0111 0010 0010 0111 1101 0010 0100 1

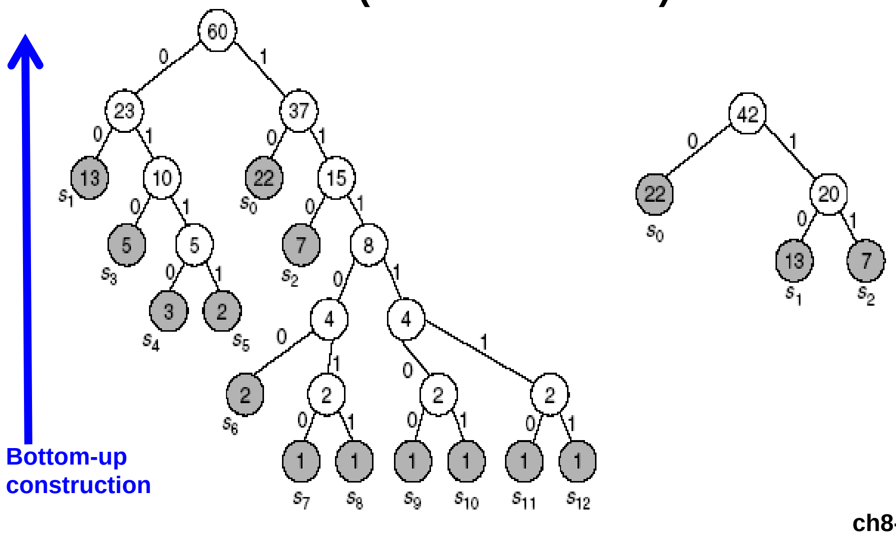
Each is code from ATE

A test vector is partitioned into a number of 4-bit patterns

Huffman Tree

(More Frequent Symbol, Shorter Code)

- Huffman code (fixed-to-variable)

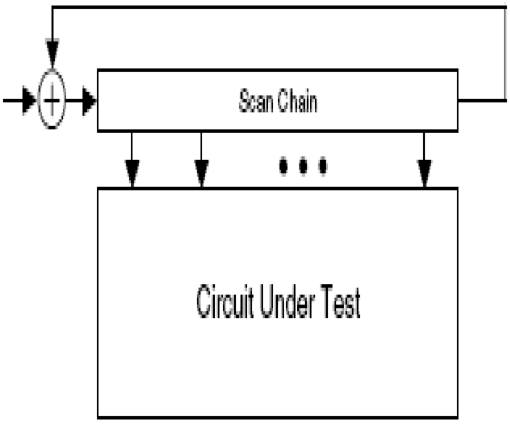


ch8-11

Run-Length Code

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

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



ch8-12

Golomb Code

- 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
...	...	...	...	...

Example of Golomb Code

- 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

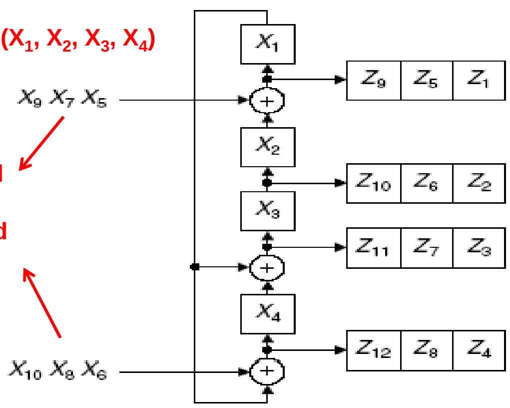
Test Stimulus Compression

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

Linear-Decompression-Based Schemes

Seed of LFSR: (X_1, X_2, X_3, X_4)

Compressed
Test vector
To be applied
From ATE



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

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

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

Matrix Form (Linear-Decompression-Based Schemes)

Decompressor Matrix (?)

Compressed Test Vector + Seed (?)

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

Original Test Vector (Given)

Solving Linear Decompressor & Its Seed

$Z = 1\text{--}011\text{-----}0$ **(Z is Test Vector)**

$X = 0111000001$

$$
\left[\begin{array}{cccccccccc|c}
0&1&0&0&1&0&0&0&0&0&1\\
1&0&0&0&0&1&0&0&0&0&0\\
0&0&1&0&0&0&1&0&0&0&1\\
1&0&0&1&0&0&0&0&0&0&1\\
0&0&1&0&0&0&1&0&0&1&0
\end{array}\right]
\xrightarrow{\text{Gaussian Elimination}}
\left[\begin{array}{cccccccccc|c}
1&0&0&0&0&1&0&0&0&0&0\\
0&1&0&0&1&0&0&0&0&0&1\\
0&0&1&0&0&0&1&0&0&0&1\\
0&0&0&1&0&1&0&0&0&0&1\\
0&0&0&0&0&0&0&0&0&1&1
\end{array}\right]
$$

(Z Vector with only care bits) **Pivot elements indicated in circles**

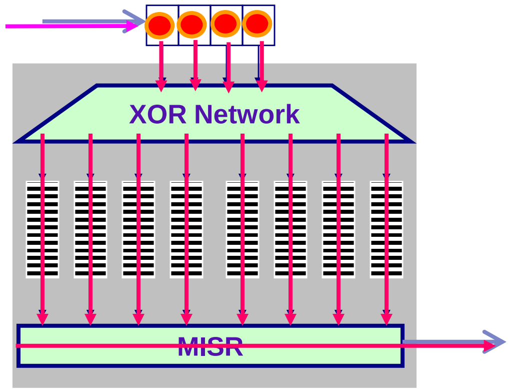
$Z = 1\text{-}0\text{--}1\text{------}$

$X =$ No Solution

$$
\left[\begin{array}{cccccccccc|c}
0&1&0&0&1&0&0&0&0&0&1\\
1&0&0&1&0&0&0&0&0&0&0\\
1&0&0&1&0&0&0&0&0&0&1
\end{array}\right]
\xrightarrow{\text{Gaussian Elimination}}
\left[\begin{array}{cccccccccc|c}
1&0&0&1&0&0&0&0&0&0&0\\
0&1&0&0&1&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0&0&0&1
\end{array}\right]
$$

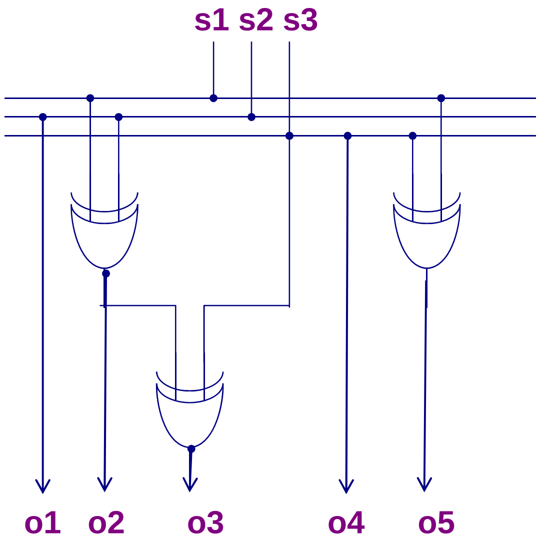
Hardware for Linear-Decompressor

XOR
Network



ch8-19

XOR Network: a 3-to-5 Example



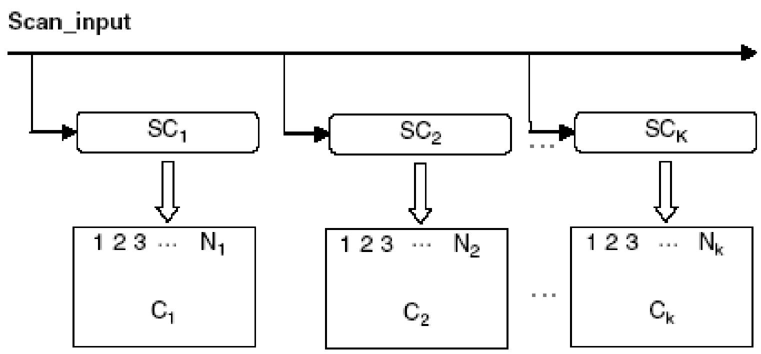
ch8-20

Test Stimulus Compression

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

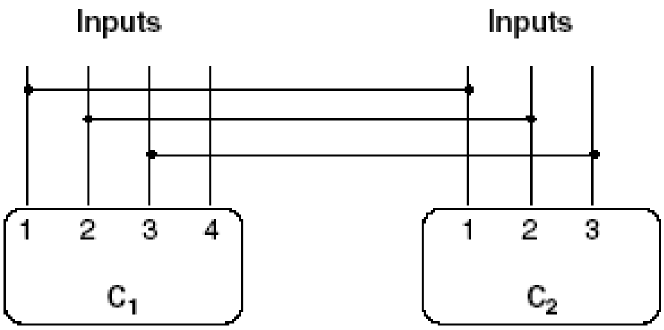
Basic Concept: Broadcast-Scan

{SC_1, SC_2, …, SC_k} shares the same test patterns applied by ATE



ATPG Supporting Broadcast-Scan

- Force ATPG tool to generate patterns for broadcast scan (by binding certain PI's together)



Reconfigurable Broadcast Scan

- Reconfigurable broadcast scan
  - 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 Scheme

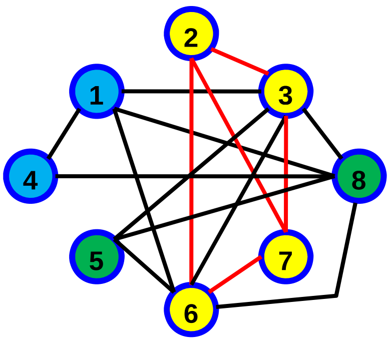
- First configuration is: 1->{2,3,6}, 2->{7}, 3->{5,8}, 4->{1,4}
- Second configuration is: 1->{1,6}, 2->{2,4}, 3->{3,5,7,8}

	First Partition					Second Partition				
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

Compatibility Graph – Finding Cliques

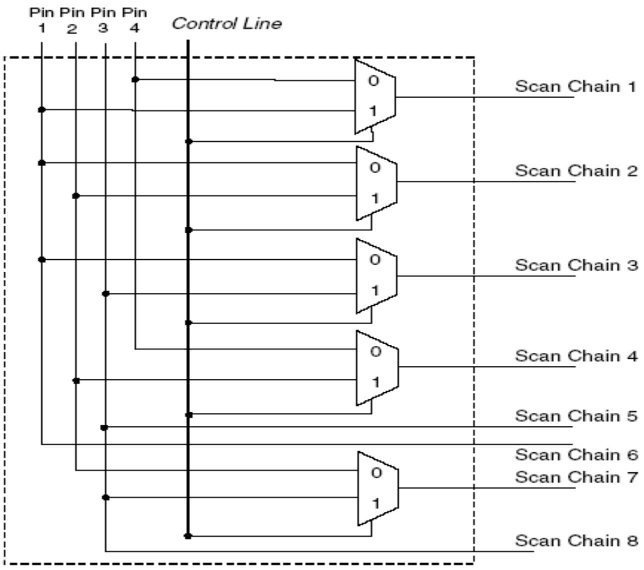
Original Test Pattern

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



Cliques (fully connected sub-graphs):
(1){SC2, SC3, SC6, SC7} → (000X1)
(2){SC5, SC8} → (0X1XX)
(3){SC1, SC4} → (111X0)
➔ Overall 8 sub-patterns down to 3

Broadcast-Scan Based Scheme

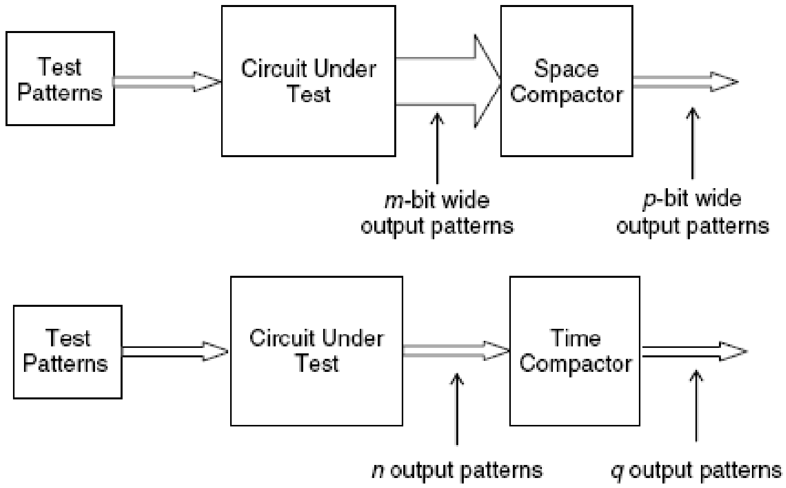


Test Response Compaction (or Called Output Compaction)

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

Unlike lossless input stimulus compression,
Output compaction is often lossy, leading to aliasing…

Test Response Compaction



ch8-29

Space (Output) Compaction

- Space (output) compaction
  - Zero-aliasing output compaction
  - X-compactor
  - X-blocking & X-masking techniques
  - X-impact-aware ATPG

ch8-30

Zero-Aliasing Output Compaction

Theorem 6.1

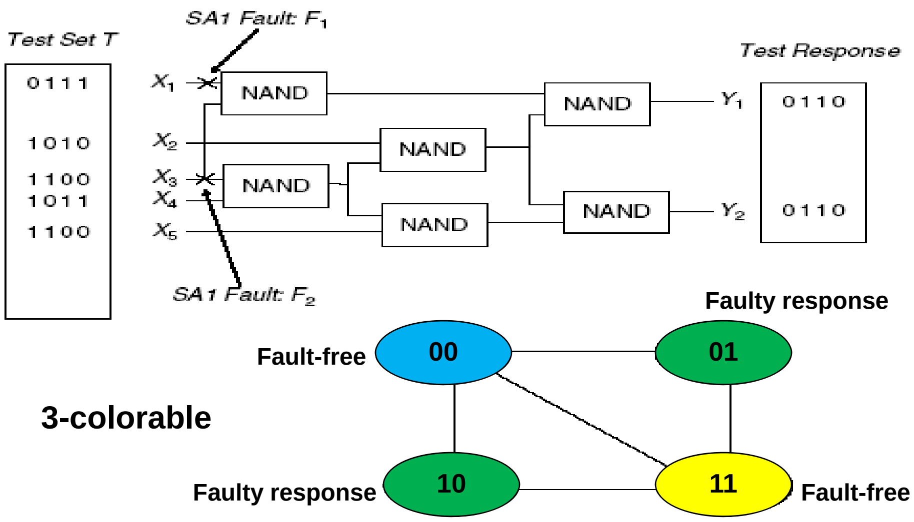
For any test set T, for a circuit that implements function C, there exists a zero-aliasing output space compactor for C with q outputs where $q = \lceil \log_2(|T|+1) \rceil$.

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.

ch8-31

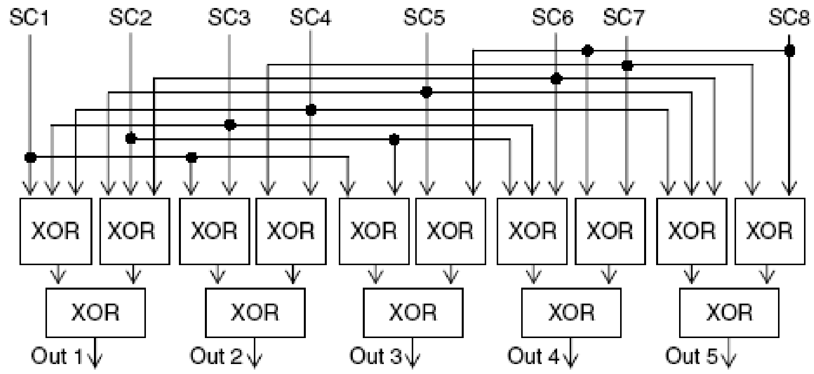
Example: Response Graph



ch8-32

Architecture of X-Compactor

- X-compactor with 8 inputs and 5 outputs



ch8-33

X-compact Matrix

Out1 Out2 Out3 Out4 Out5

$$M = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix} \begin{matrix} \text{SC1} \\ \text{SC2} \\ \text{SC3} \\ \text{SC4} \\ \text{SC5} \\ \text{SC6} \\ \text{SC7} \\ \text{SC8} \end{matrix}$$

→ **SC1 drives {Out1, Out2, Out3}**

Matrix Form:

$$SC_{1x8} \cdot M_{8x5} = \begin{bmatrix} Out1 \\ Out2 \\ Out3 \\ Out4 \\ Out5 \end{bmatrix} = Out_{5x1}$$

$$SC = [SC1 \quad SC2 \quad SC3 \quad SC4 \quad SC5 \quad SC6 \quad SC7 \quad SC8]$$

ch8-34

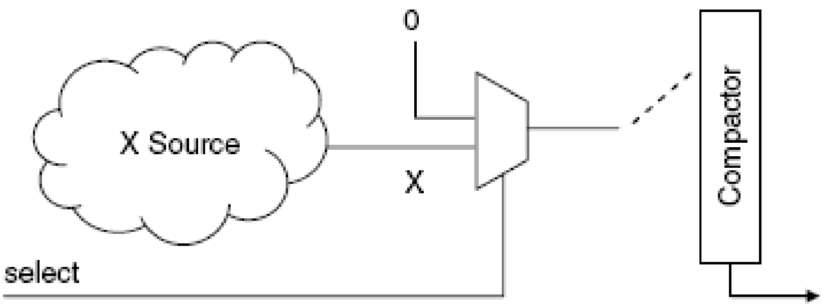
X-Blocking or Masking Techniques

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

ch8-35

X-Blocking by Selection

- Illustration of the X-blocking scheme

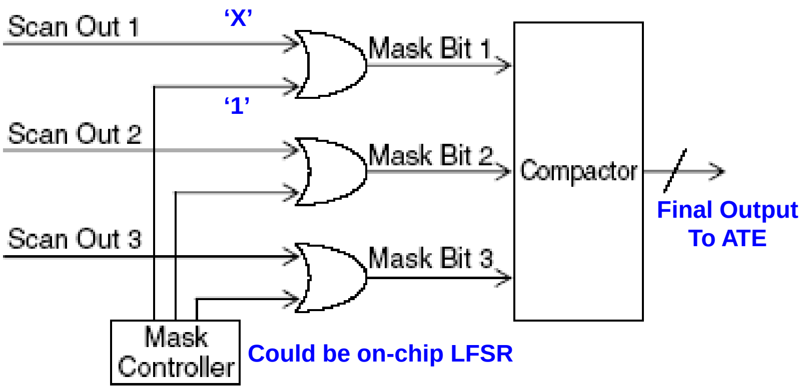


(This is a bit stream provided by ATE)

Final Output
To ATE

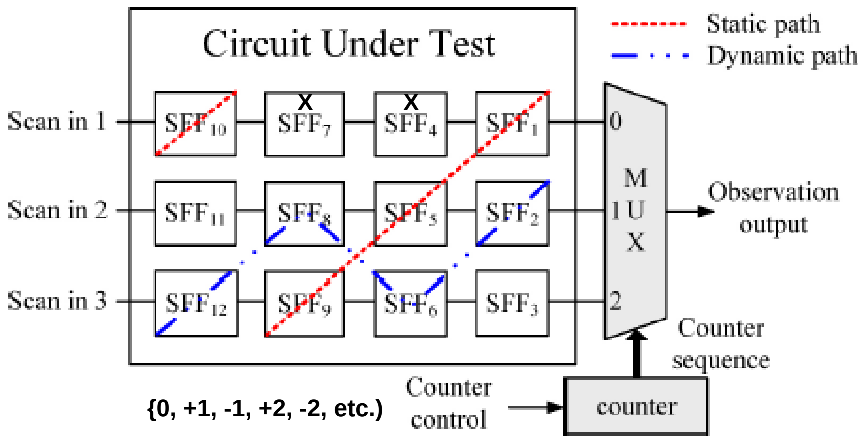
ch8-36

X-Masking by Masking Logic



When there is an X in a scan chain output,
a controlling value , i.e., '1' in this example, is issued to mask it out

X-Tolerance by Counter-Based Output Selection



Dynamic path means counter operation can be changed at any scan cycle

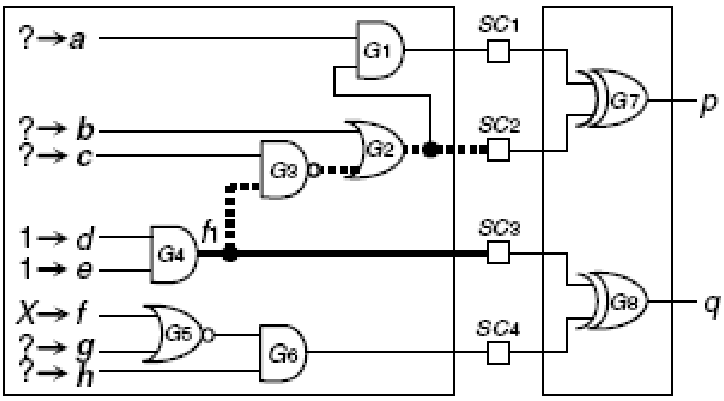
X-Impact-Aware ATPG

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

ch8-39

Example: Handling X in ATPG

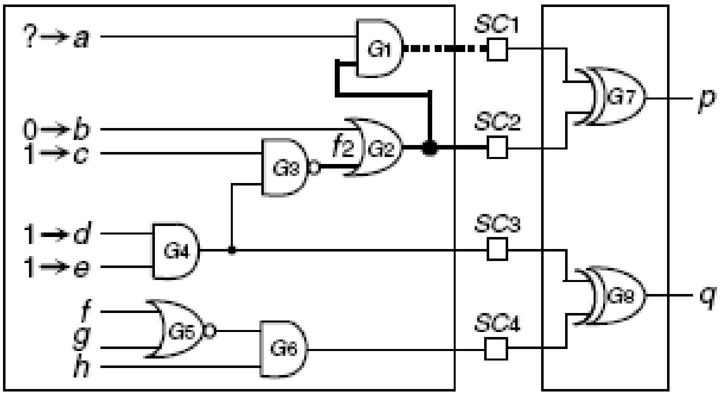
Path (G5→G6→SC4→G8→q) might be contaminated by 'X' at f
(1) Propagate the fault effect through (f1→G3→G2→SC2→G7→p) → b=0, c=1
(2) Kill the X by assigning g to '1' → SC4=0 → q is observable



ch8-40

Output-Compactor-Aware ATPG

- f_2/1 fault could be masked as propagated to *p*
- Block aliasing by assigning *a* to '0'



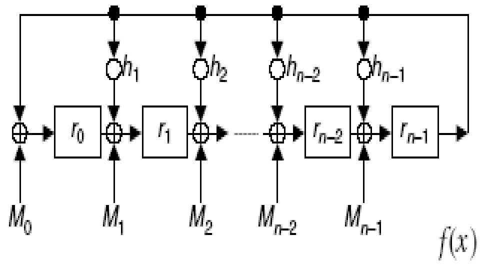
ch8-41

Time Compaction

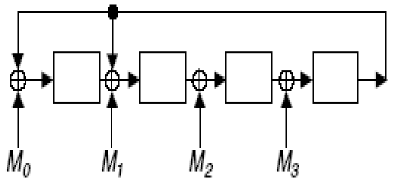
- Time compaction
  - A time compactor uses sequential logic to compact test responses
  - MISR is most widely adopted
  - n-stage MISR can be described by specifying a *characteristic polynomial* of degree *n*

ch8-42

Multiple-Input Signature Register



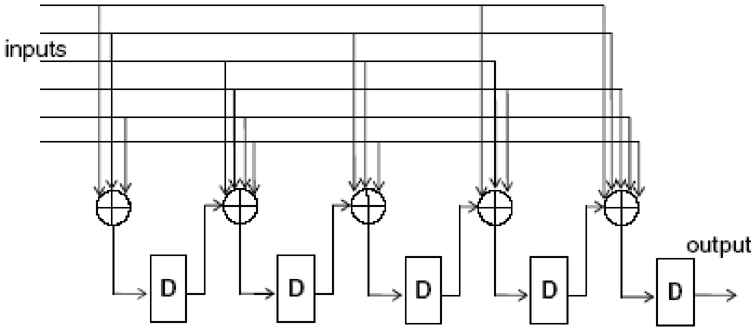
$$f(x) = 1 + h_1x + h_2x^2 + \cdots + h_{n-1}x^{n-1} + x^n$$



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

Mixed Time & Space Compaction

- Mixed time and space compaction



Industry Practices

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

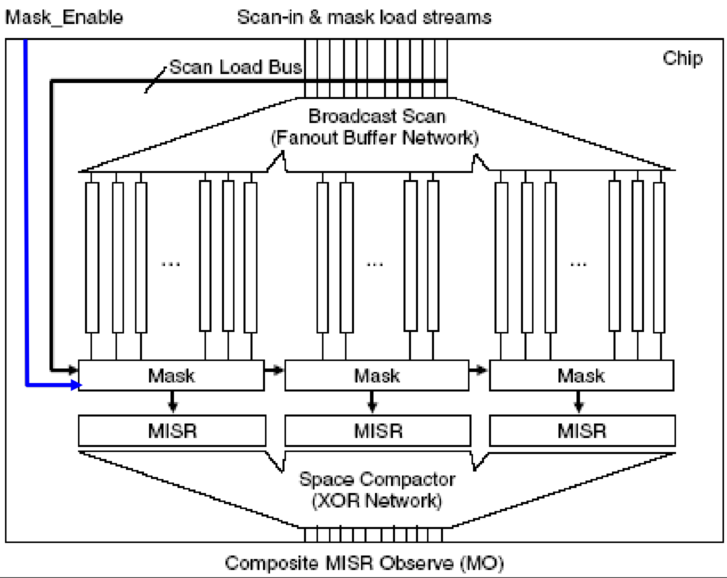
ch8-45

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

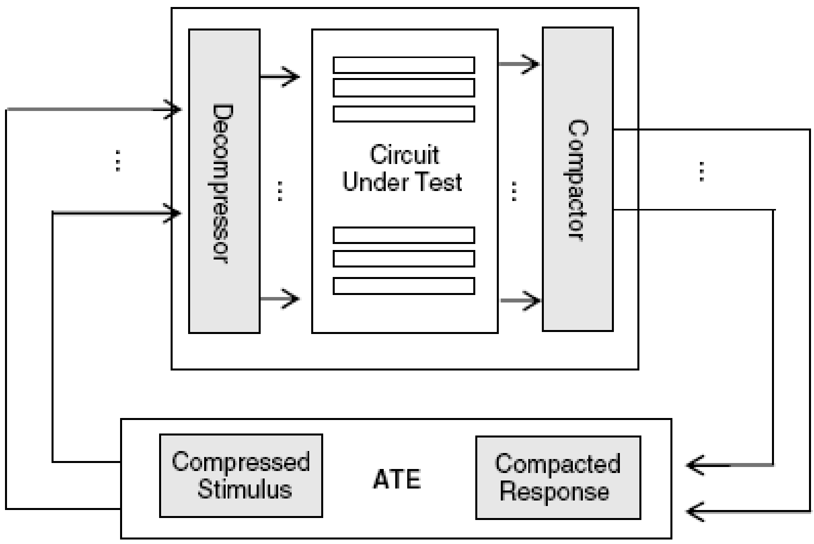
ch8-46

General Scan Architecture for OPMISR+



ch8-47

EDT (TestKompression) Architecture



ch8-48

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**
  - **Successful as part of standard design flow**