Test Compression

- Introduction
- Software Techniques
- Hardware Techniques
  - Test Stimulus Compression
  - Test Response Compression
- Industry Practices
- Conclusion

More than 1000x Compression Needed

Year of Production	2013	2014	2015	2016	2017	2018	2019	2020
Worst Case (Flat) Data Volume (Gb)								
MPU-HP - High Performance MPU (Server)	1458	1984	2699	3673	4998	6138	7537	9256
MPU-CP - Consumer MPU (Laptop/Desktop)	853	1160	1579	2149	2924	3591	4409	5415
SOC-CP - Consumer SOC (Consumer SOC, APU, Mobile Proces	1122	1526	2077	2826	3846	4723	5800	7122
Best-Case Test Data Volume (Hierarchal & Compression) (Gb)								
MPU-HP - High Performance MPU (Server)	4.7	5.1	5.7	6.4	7.2	7.3	7.4	7.5
MPU-CP - Consumer MPU (Laptop/Desktop)	3.7	4.1	4.6	5.1	5.7	5.7	5.8	5.8
SOC-CP - Consumer SOC (Consumer SOC, APU, Mobile Proces	6.9	7.9	8.8	10.2	11.6	12.2	12.6	12.5
Best-Case Compression Ratio (Hierarchal & Compression)								
MPU-HP - High Performance MPU (Server)	312	389	471	572	694	842	1022	1242
MPU-CP - Consumer MPU (Laptop/Desktop)	231	280	342	425	516	625	758	926
SOC-CP - Consumer SOC (Consumer SOC, APU, Mobile Proces	162	192	236	278	330	388	461	568

Review: Real BIST Case (see 14.6)

Item	ASIC1	ASIC2	ASIC3	ASIC4
Gate count	180K	356K	550K	748K
BIST pattern count	65K	262K	262K	262K
ATPG pattern count	2.5K	17K	13K	20K
BIST SSF Cov.(%)	96.0	95.7	95.3	95.6
ATPG SSF Cov.(%)	97.8	97.8	97.2	97.9
BIST frequency (MHz)	125	75	75	75
ATPG frequency	50	40	20	50
BIST test time (sec.)	0.06	0.58	0.93	1.2
ATPG test time (sec.)	0.02	0.36	0.94	0.7
BIST volume (MB)	0	0	0	0
ATPG volume (MB)	24	344	451	828

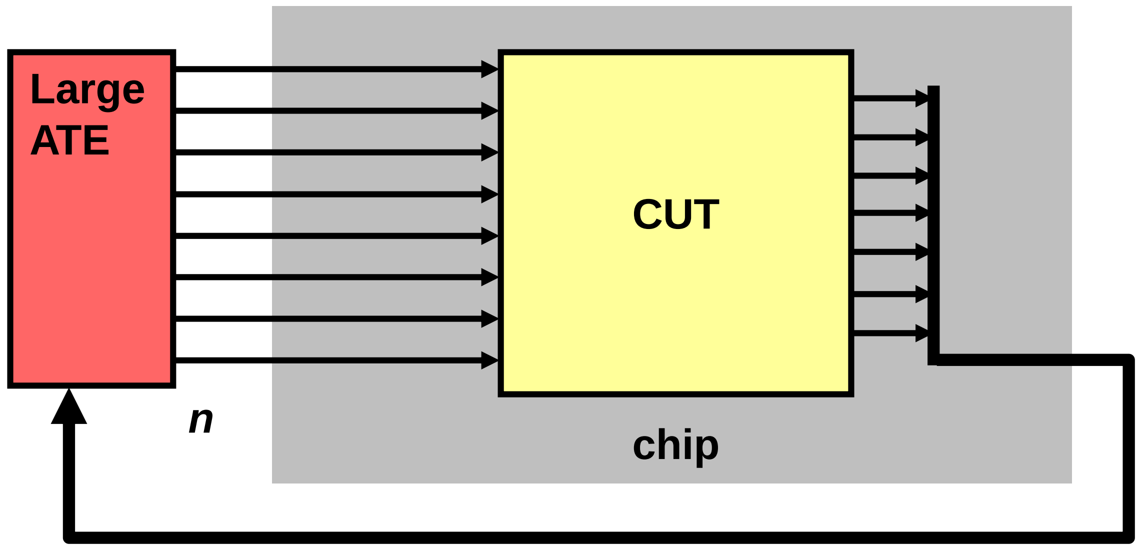
Comparison of Techniques

<u>Pure BIST</u>	<u>HW Test Compression</u>	<u>Traditional ATPG/ATE</u>
+ Low cost ATE	+ Low cost ATE	- Expensive ATE
+ Zero test data	+ Small test data	- Large test data
- Low F.C.	+ High F.C.	+ High F.C.
- No test time reduction	+ Test time reduction	- No test time reduction

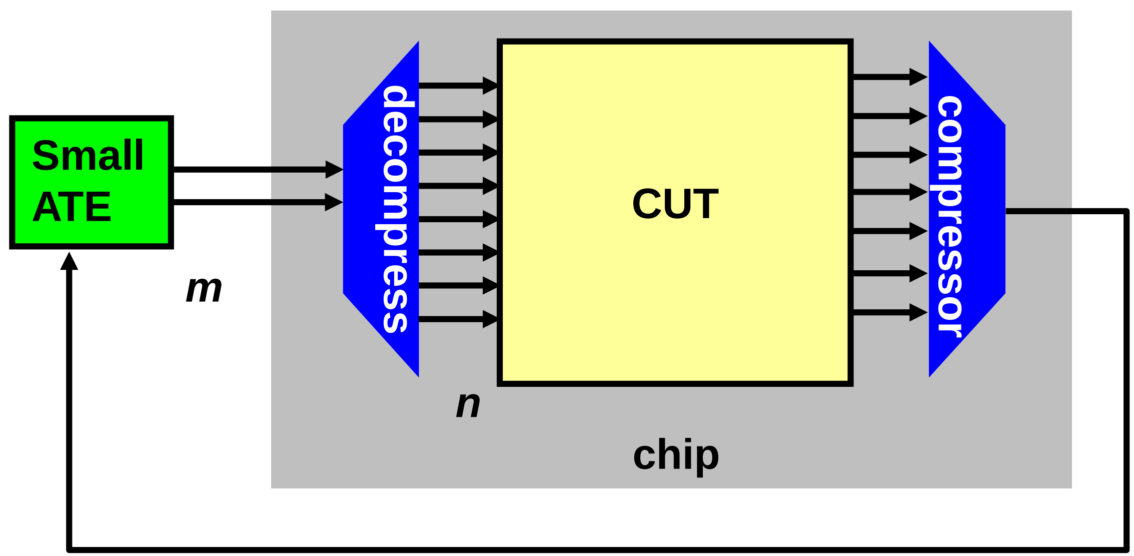
Test Compression Has Advantages of Both Sides

W/Wo Test Compression DFT

- **Without compression**
  - *n* very large

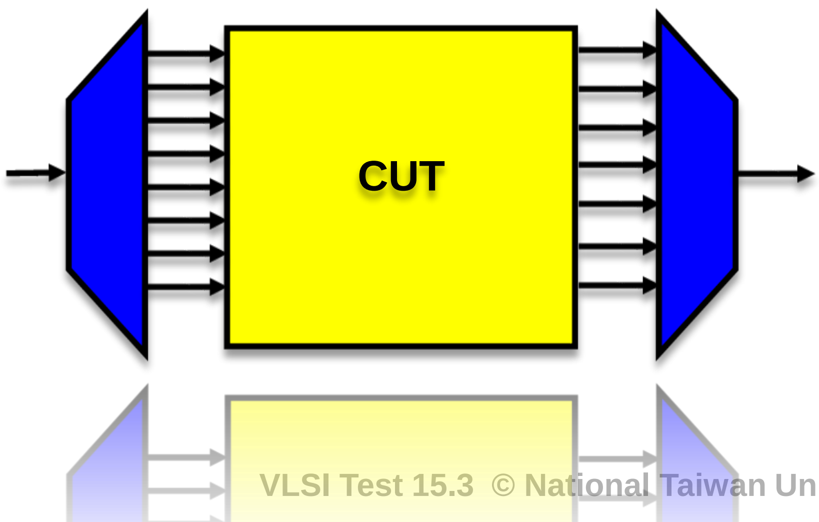


- **With compression**
  - *m* << *n*



Test Compression

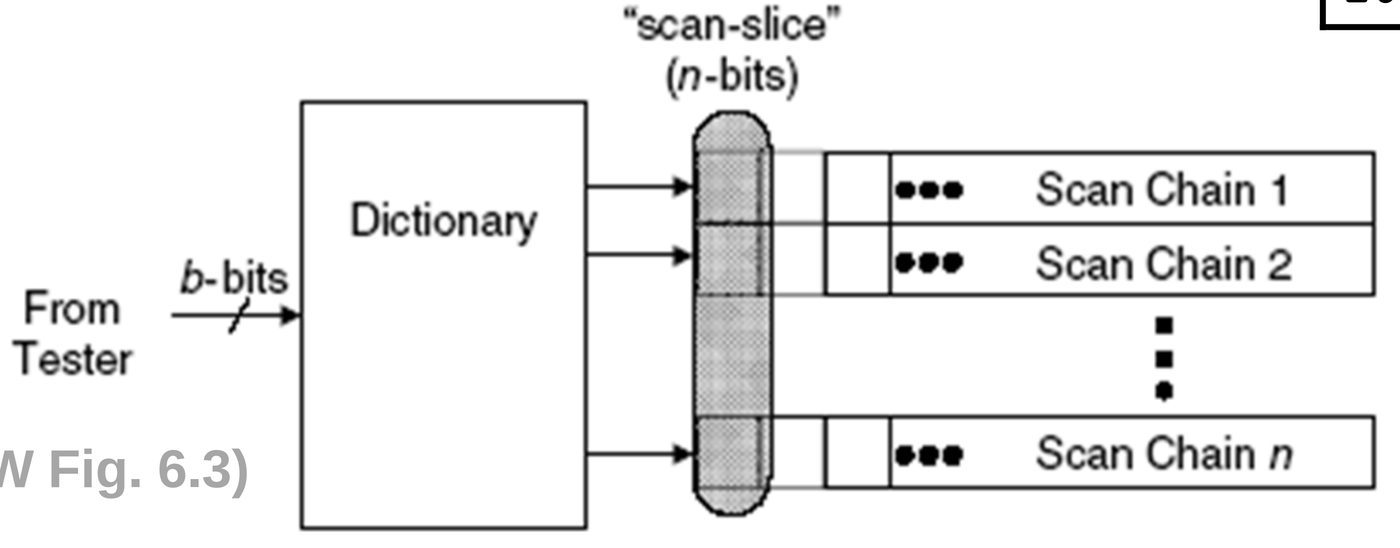
- Introduction
- Software Techniques
- Hardware Techniques
  - Test Stimulus Compression
    - Code-based schemes
      - Dictionary code (fixed-to-fixed) [Reddy 2002]
      - Huffman code (fixed-to-variable) [Jas 2003]
    - Broadcast-based schemes
    - Linear-decompression-based schemes
  - Test Response Compression
- Industry Practices
- Conclusion



Dictionary Code (fixed-to-fixed)

- Dictionary compresses a ***symbol*** into a ***codeword***
  - 2^b codewords, 2^n symbols. $n > b$
- **Fixed-to-fixed:** original data is fixed rate (=n)
  - compressed data is fixed rate (=b)
- A ***scan slice*** is a vertical column of scan data

b=2	n=4
00	0011
01	0101
11	0111
10	1110



(WWW Fig. 6.3)

$$Compression\ \ Ratio = \frac{Original\ \ Data}{Compressed\ \ Data} = \frac{n}{b}$$

Huffman Code (fixed-to-variable)

- Count the frequency of occurrence for each symbol
  - Higher frequency symbols are shorter codewords
- Fixed-to-variable: original data is fixed rate
  - compressed data is variable rate

(WWW Table 6.2)

Symbol	Frequency	Pattern	Huffman Code
S_0	22	0 0 1 0	1 0
S_1	13	0 1 0 0	0 0
S_2	7	0 1 1 0	1 1 0
S_3	5	0 1 1 1	0 1 0
S_4	3	0 0 0 0	0 1 1 0
S_5	2	1 0 0 0	0 1 1 1
S_6	2	0 1 0 1	1 1 1 0 0
S_7	1	1 0 1 1	1 1 1 0 1 0
S_8	1	1 1 0 0	1 1 1 0 1 1
S_9	1	0 0 0 1	1 1 1 1 0 0
S_{10}	1	1 1 0 1	1 1 1 1 0 1
S_{11}	1	1 1 1 1	1 1 1 1 1 0
S_{12}	1	0 0 1 1	1 1 1 1 1 1
S_{13}	0	1 1 1 0	—
S_{14}	0	1 0 1 0	—
S_{15}	0	1 0 0 1	—

Original test pattern

```
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
```

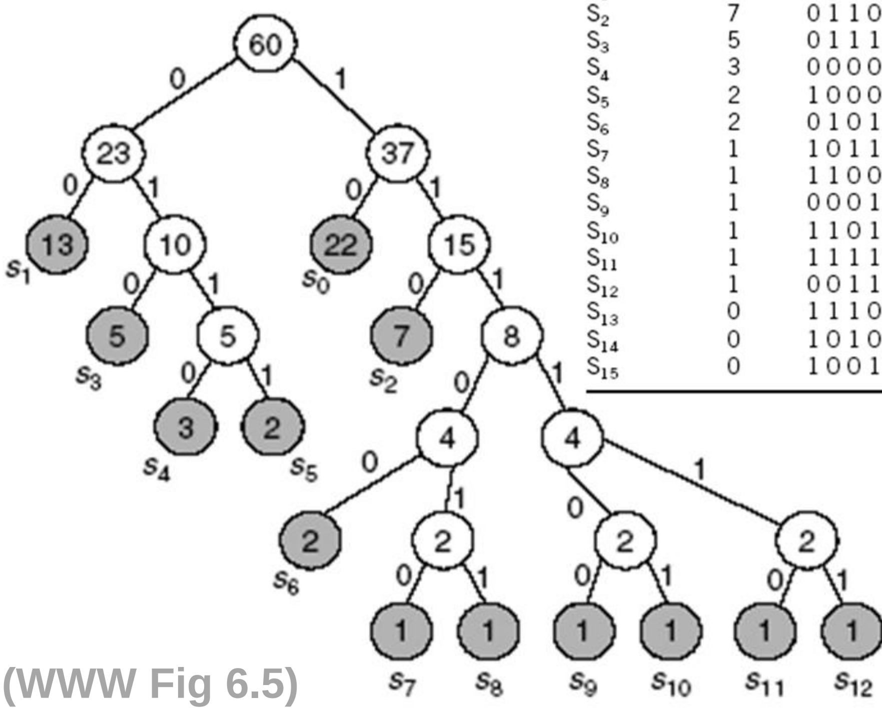
Compressed test pattern

```
10  00  10  110  0110  10  111010 00 00 00 110 10
10  00  10  110  0110  110 10  00  110 10  10  0110
10  110  10 10  10  00  110  10  10  0111  11100
111100  00 10  010  10  010  010  00  00  00  0111  11100
111011  00  00  010  10  10  110  111101  10  00  111110  11111
```

Huffman Code (2)

- **Greedy algorithm (details see [CLRS 09] Chapter 16)**
  - **Merge two lowest frequency as one node**

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



(WWW Fig 6.5)
Numbers in node = frequency

HUFFMAN (C)
n=|C|
Q=C
for i = 1 **to** n-1
  allocate a new node z
  $z.left$ = x = EXTRACT-MIN (Q)
  $z.right$= y = EXTRACT-MIN(Q)
  $z.freq$=$x.freq$+$y.freq$
  INSERT(Q,z)
return EXTRACT-MIN(Q)

Huffman is Optimal Prefix Code

QUIZ

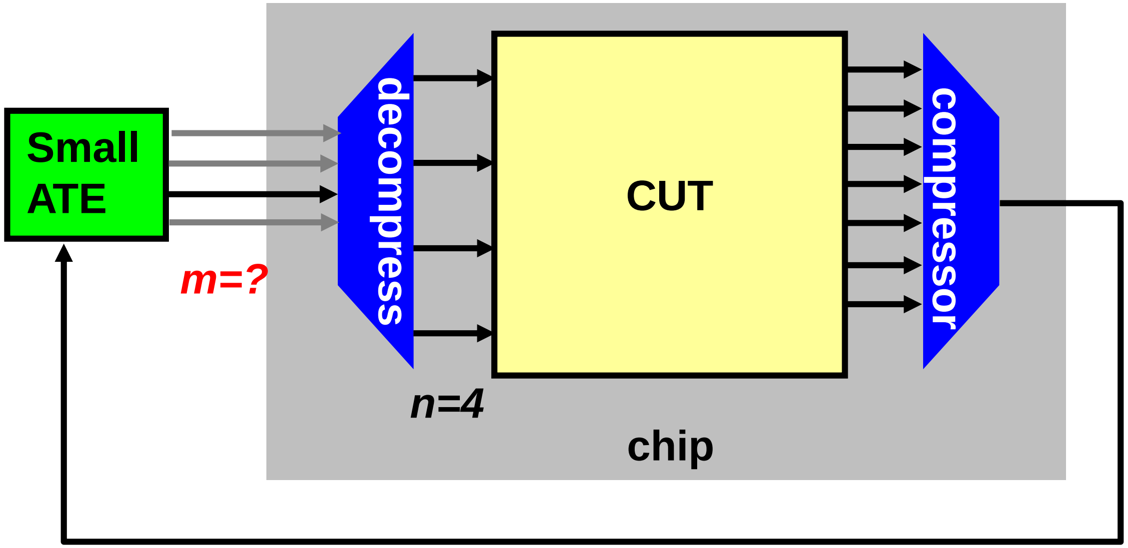
Q: What is compression ratio of this Huffman code?

Frequency	**Pattern**	**Huffman code**
6	0000	1
3	0100	00
1	1010	01

ANS:

Problems with Code-based Schemes

- Dictionary too large
  - Hardware overhead
- Synchronization problem (Huffman)
  - ATE sends data at fixed rate, but Huffman require variable rate
  - Not easy to implement



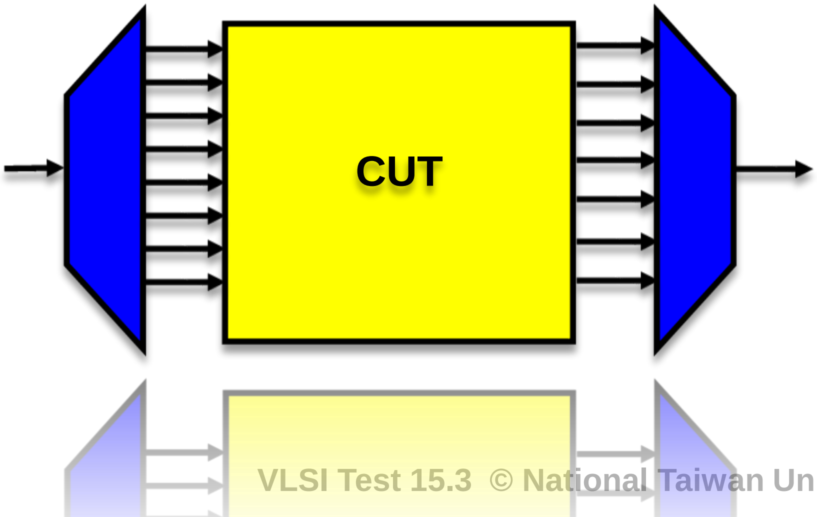
n=4 m=variable

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

Code-based Not Useful in Practice

Test Compression

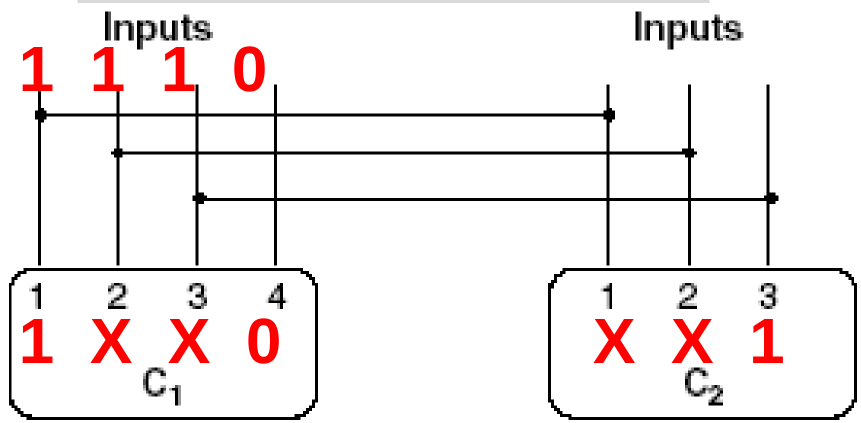
- Introduction
- Software Techniques
- Hardware Techniques
  - Test Stimulus Compression
    - Code-based schemes
    - Broadcast-based schemes
    - Linear-decompression-based schemes
  - Test Response Compression
- Industry Practices
- Conclusion



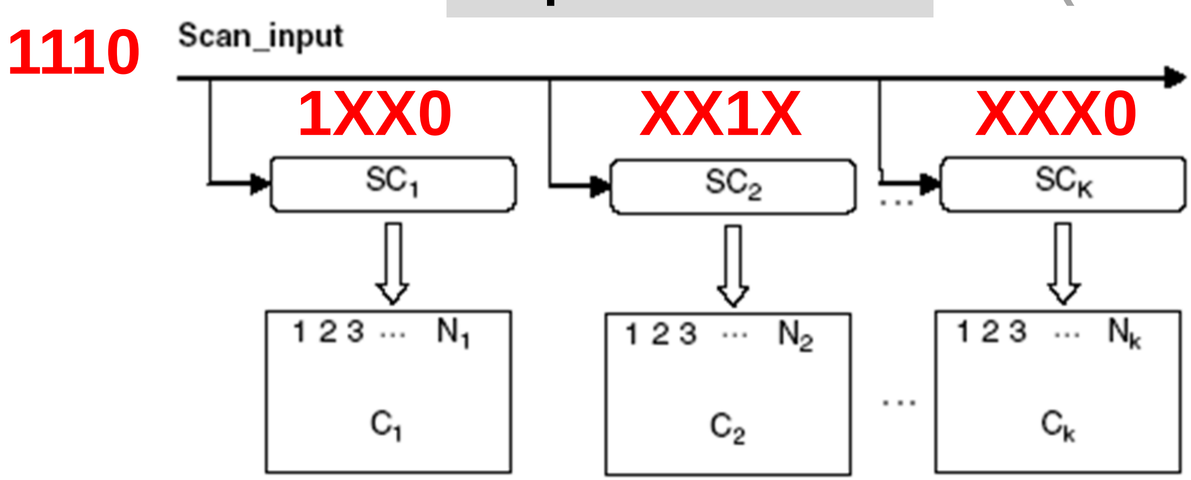
Broadcast Scan [Lee 1998]

- **Many test patterns are compatible. Just broadcast them!**

combinational circuit (WWW Fig. 6.16)



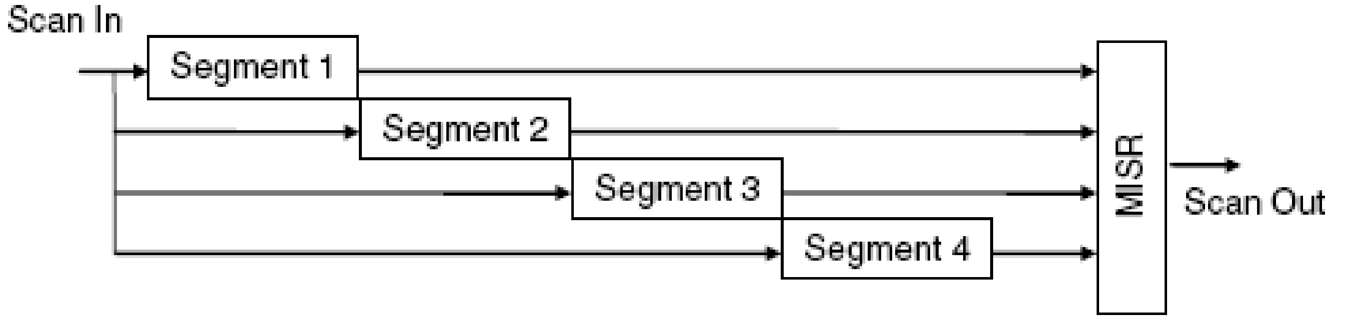
sequential circuit (WWW Fig. 6.17)



Illinois Scan [Hamzaoglu 99]

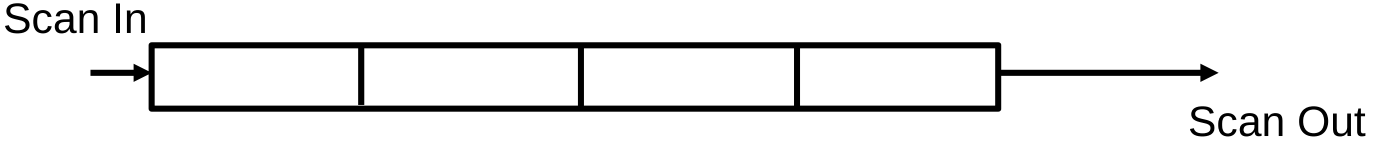
- **First apply compatible test patterns in *broadcast mode***
  - **then apply incompatible test patterns in *serial mode***

broadcast mode



Serial mode

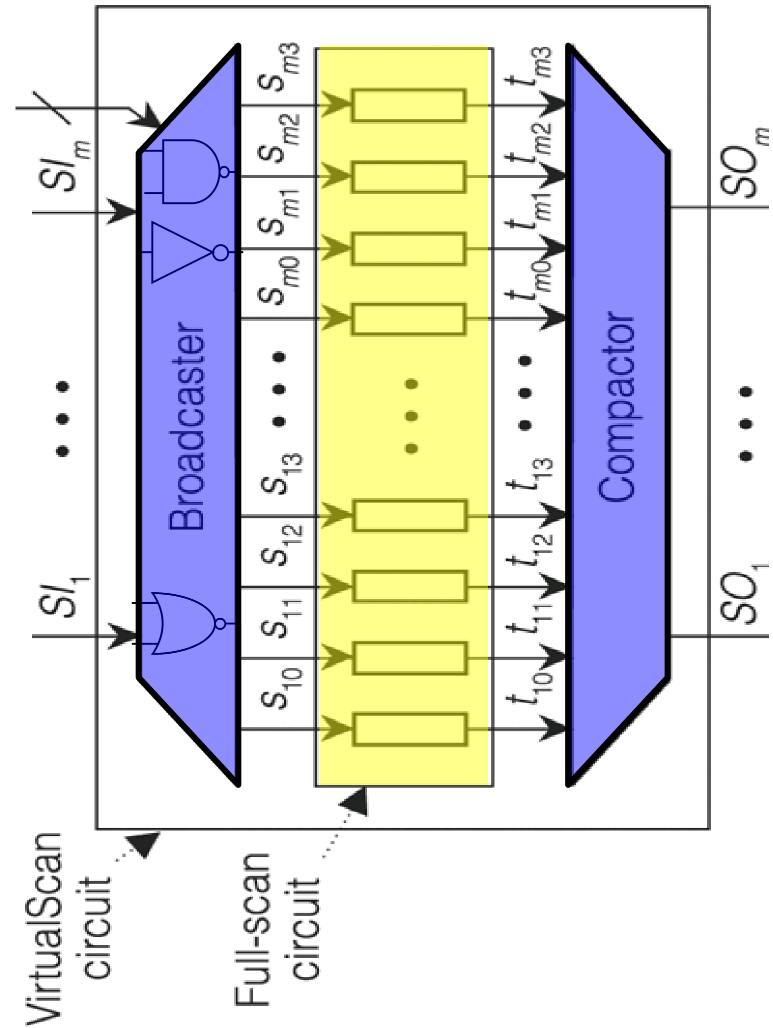
(WWW Fig. 6.19)



Virtual Scan [Wang 2002]

- ***Broadcaster*** **consists of gates:**
  - **XOR, INV, MUX, AND/OR...**

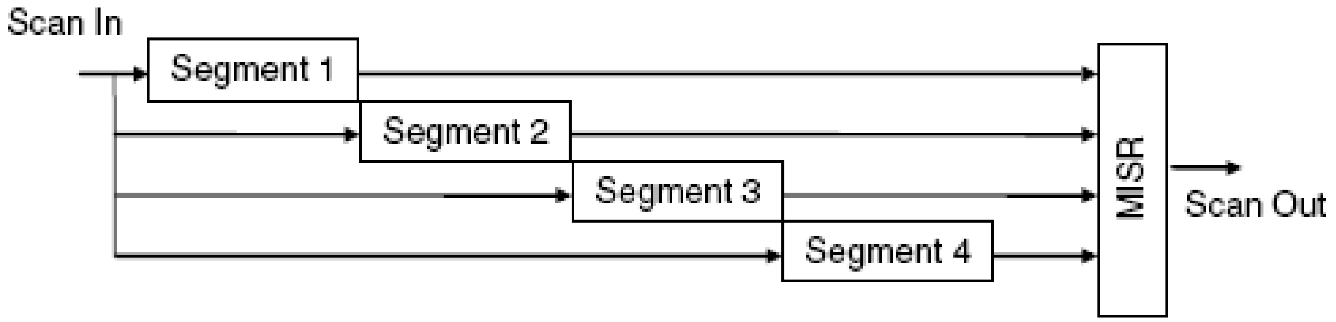
- **When ATPG, broadcaster is treated as CUT**
  - **No need to solve linear equations**
  - **Dynamic compression can be effectively utilized during ATPG**
  - **Very little fault coverage loss**



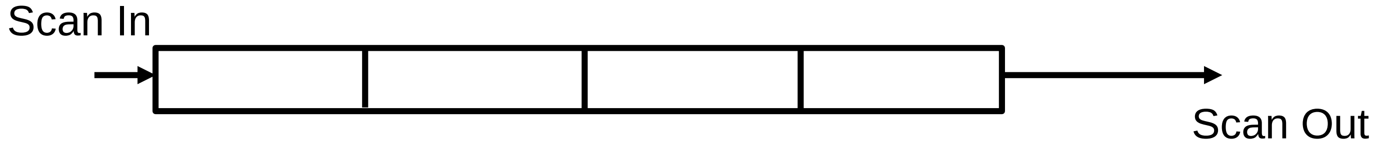
QUIZ

Q: Suppose 95% test patterns are in broadcast mode, the other 5% are in serial mode. What is compression ratio?
ANS:

broadcast mode

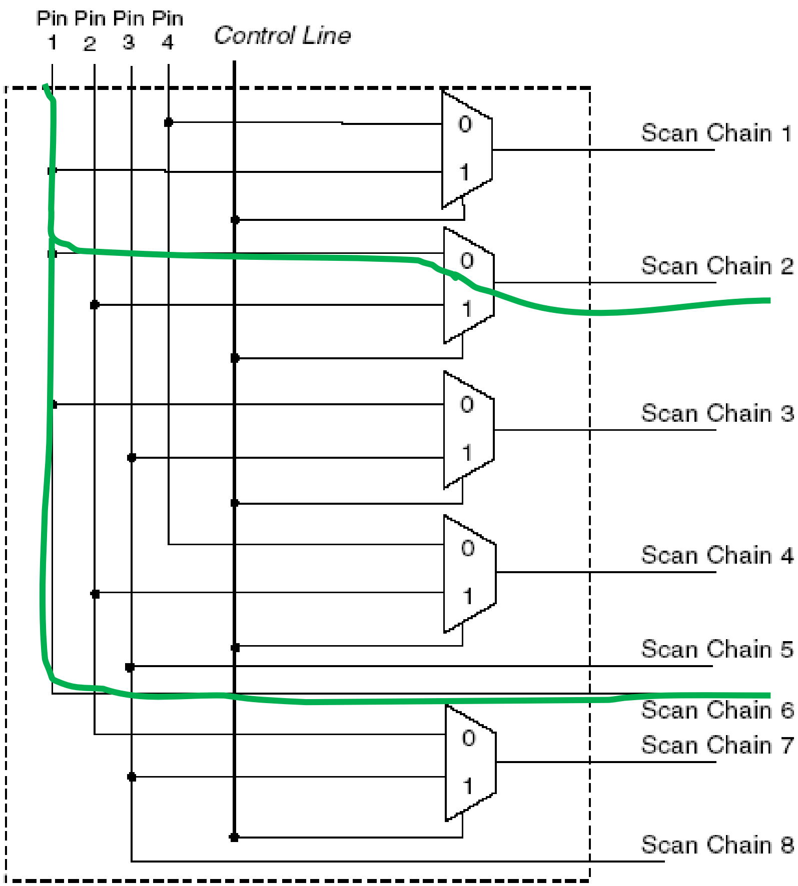


Serial mode



How about Incompatible Test Patterns?

- *Reconfigurable* **broadcaster**
  - **Control=0: 1→{2,3,6}, 2→{7}, 3→{5,8}, 4→{1,4}**
  - **Control=1: 1→{1,6}, 2→{2,4}, 3→{3,5,7,8}**

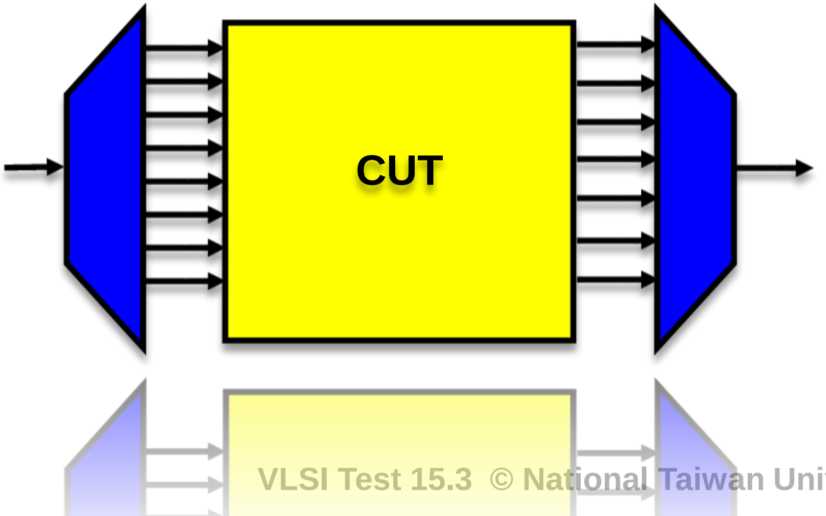


	2,6 compatible					2,6 incompatible				
Scan Chain 1	1	X	1	X	X	X	0	0	X	X
Scan Chain 2	X	X	0	X	1	0	X	1	X	1
Scan Chain 3	X	X	X	X	1	1	1	X	X	1
Scan Chain 4	1	1	X	X	0	0	0	X	0	1
Scan Chain 5	0	X	1	X	X	X	X	X	X	X
Scan Chain 6	X	0	X	1	X	0	X	0	0	X
Scan Chain 7	0	X	0	X	X	1	1	X	X	X
Scan Chain 8	X	X	1	X	X	X	X	1	X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

(WWW Fig. 6.20)

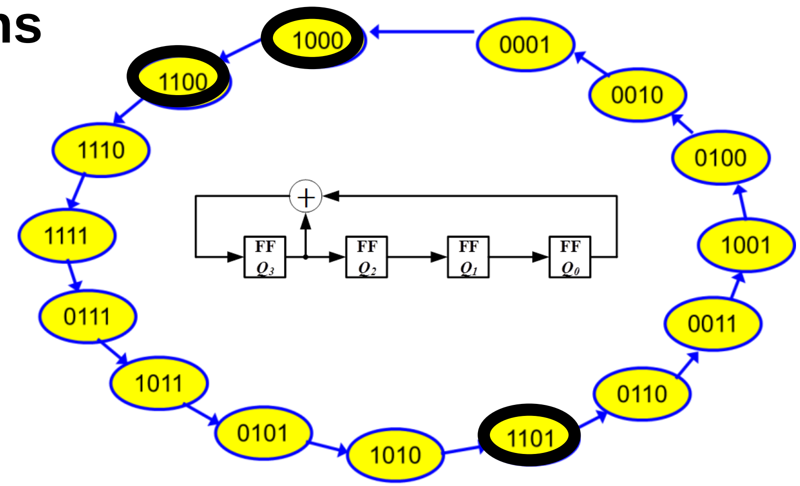
Test Compression

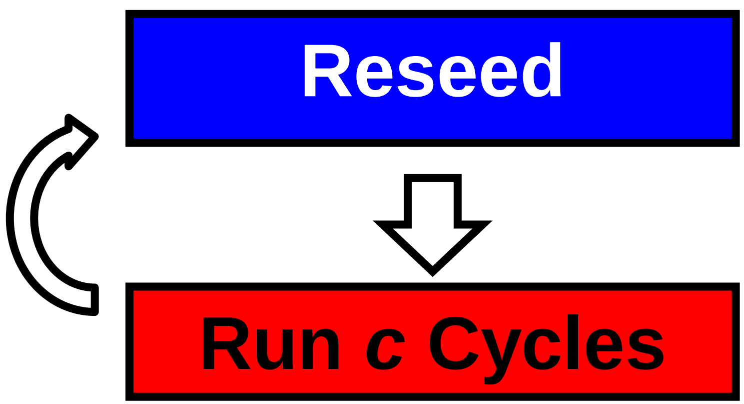
- Introduction
- Software Techniques
- Hardware Techniques
  - Test Stimulus Compression
    - Code-based schemes
    - Broadcast-based schemes
    - Linear-decompression-based schemes
  - Test Response Compression
- Industry Practices
- Conclusion

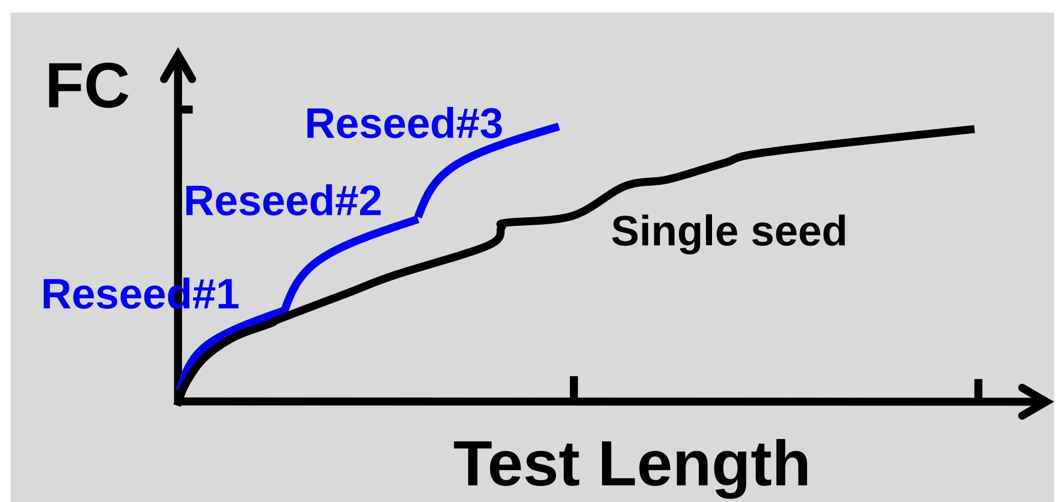


Review: LFSR Reseeding [Könemann 91]

- Load LFSR seed, run *c* cycles, reseed, run *c* cycles, …
- Example: want three patterns ‘1000, 1100, 1101’
  - Initial seed ‘1000’, apply *c*=2 patterns
  - reseed ‘1101’, apply c=2 patterns
  - Only 4 cycles

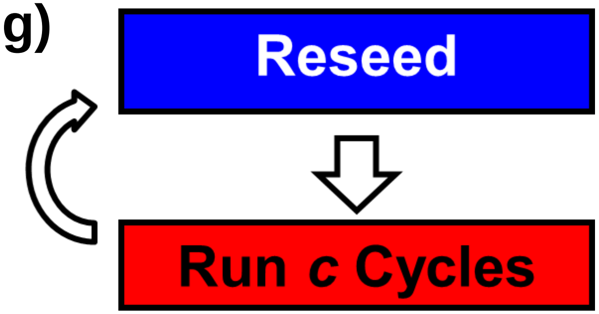






LFSR Reseeding (2)

- Reseeding also useful for **test data compression**
- Example: *N*-degree LFSR
  - Chain length L=1000 (assume 5% are care bits)
  - c =10 (apply 10 patterns between reseeding)
  - Compression ratio = **143**



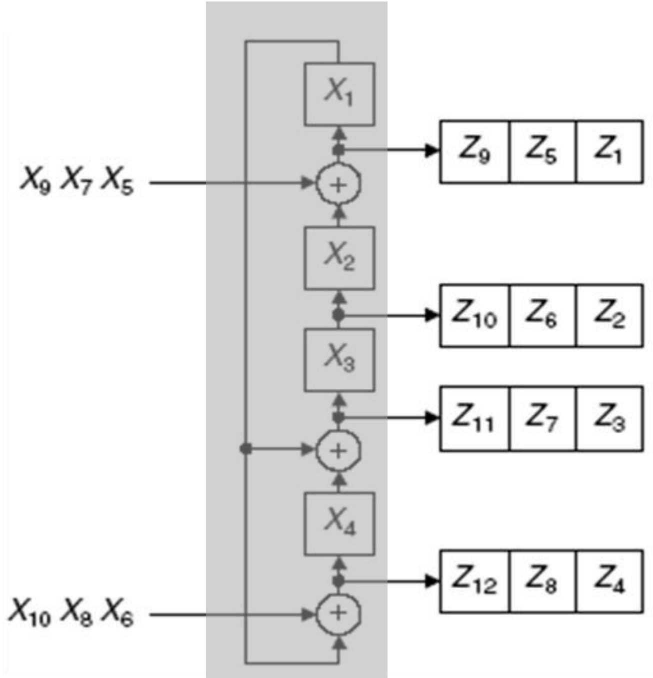
$$N = care\ bits + 20 = 1000 \times 5\% + 20 = 70 \quad \text{(see 13.4)}$$

$$Compression\ \ Ratio = \frac{Original\ \ Data}{LFSR\ \ Seed} = \frac{L \times c}{N} = \frac{10000}{70} = 143$$

Reseeding Very Good CR.
But How to Improve FC?

Ring Generator [Mrugalski 03]

- LFSR with **multiple external inputs**
  - 4 scan chains, 2 external inputs. CR = 2
- Run ATPG on CUT, then solve linear equations
  - 12 equations, 10 variables



(WWW Fig. 6.9)

More Variables
Better FC

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

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

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

Solving Ring Gen. Inputs

- **Similar to LFSR seed solving**
  - **see 13.3**
- **Solve *AX=Z***
  - ***A* has *R* rows, *C* columns**
    - ***R*=12, *C*=10**
  - ***X* is free variables**
  - ***Z* is desired test patterns**

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

- **Gauss-Jordan Elimination**
  - **O(CR^2)**

$Z = $ 1--011-----0 $\qquad\qquad$ $X = $ 0111000001

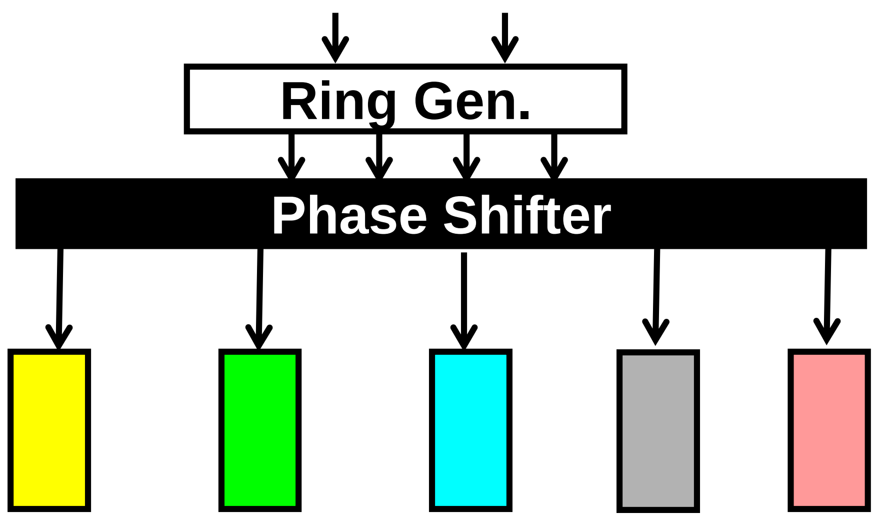
$$
\left(\begin{array}{cccccccccc|c}
0&1&0&0&1&0&0&0&0&0&1\\
1&0&0&0&0&1&0&0&0&0&0\\
0&0&1&0&0&0&1&0&0&0&1\\
1&0&0&1&0&0&0&0&0&0&1\\
0&0&1&0&0&0&1&0&0&1&0
\end{array}\right)
\xrightarrow[\text{Elimination}]{\text{Gaussian}}
\left(\begin{array}{cccccccccc|c}
1&0&0&0&0&1&0&0&0&0&0\\
0&1&0&0&1&0&0&0&0&0&1\\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

$Z = $ 1-0--1------ $\qquad\qquad$ $X = $ No Solution

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

Still No Solution?

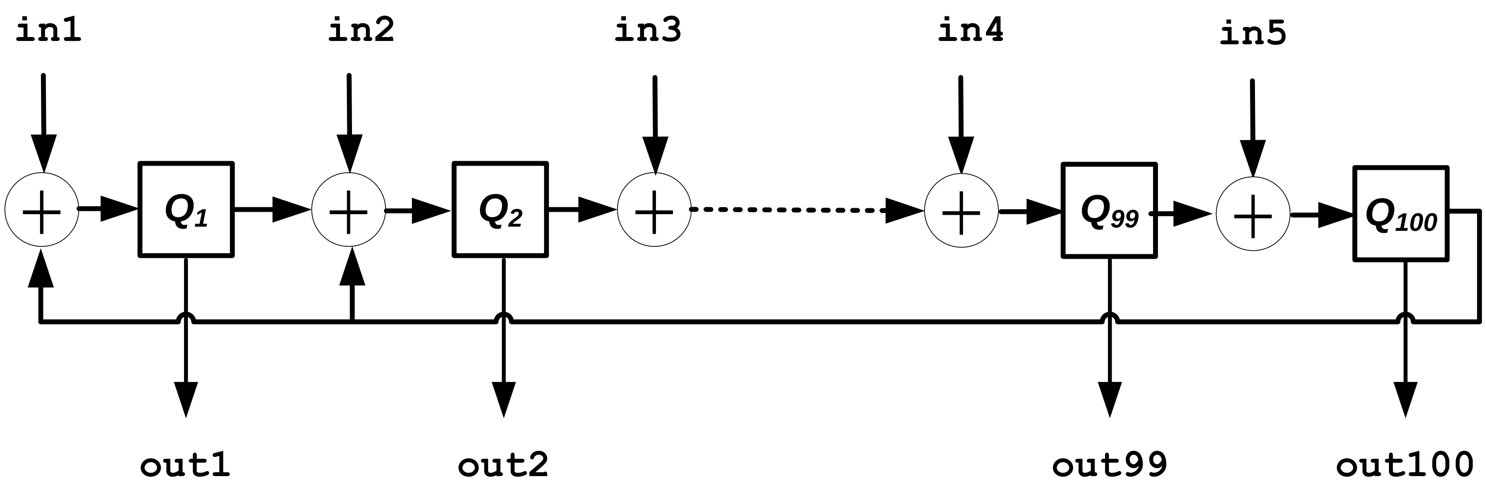
- **Remedies**
  1. **Bypass de-compressor**
     Increase data volume
  2. **Rerun ATPG**
     Increase run time
  3. **Redesign linear de-compressor**
     Increase free variables
  4. **Phase shifter**
     see 14.5



QUIZ

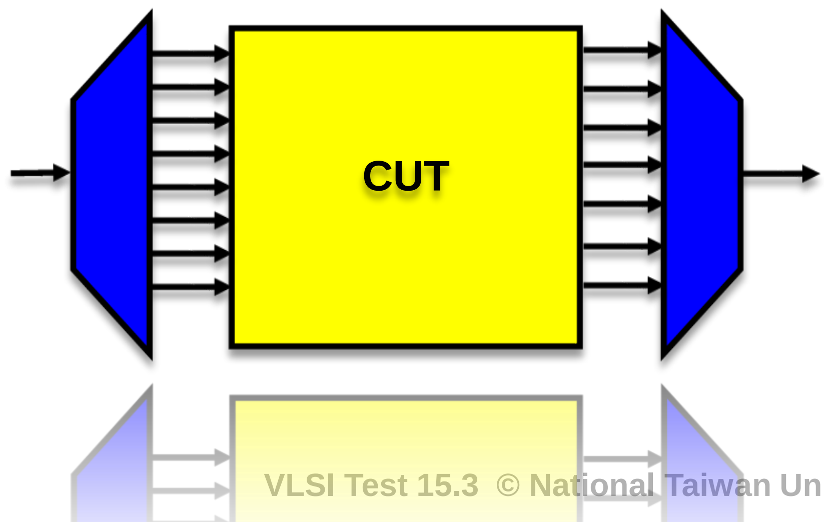
Q: What is compression ratio of this ring generator?
5 inputs, 100 outputs

ANS:



Summary

- Test Stimulus Compression
  - Code-based schemes
    - Dictionary code , Huffman code
  - Broadcast-based schemes
    - Simple and effective
  - Linear-decompression-based schemes
    - Very good CR
    - Most widely used technique



FFT

- We know ring generator has very good CR.
  - Can we use it to reduce test time?