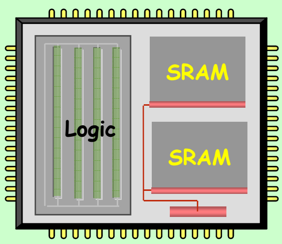
國立清華大學電機系

EE-6250
超大型積體電路測試
VLSI Testing

Chapter 7
Built-In Self-Test

Design-for-Testability

- Design activities for generating a set of test patterns with a high fault coverage.
- Methodology
  - Logic
    - Automatic Test Pattern Generation (ATPG)
    - Scan Insertion (to ease the ATPG process)
    - Built-In Self-Test
  - Memory (SRAM, DRAM, …)
    - Built-In Self-Test



Outline

- Basics
- Test Pattern Generation
- Response Analyzers
- BIST Examples
- Memory BIST

Definition & Advantages of BIST

- Built-In Self-Test (BIST) is a design-for-testability (DFT) technique in which testing (test generation , test application) is accomplished through built-in hardware features.
  - [V.D. Agrawal, C.R. Kime, and K.K. Saluja]

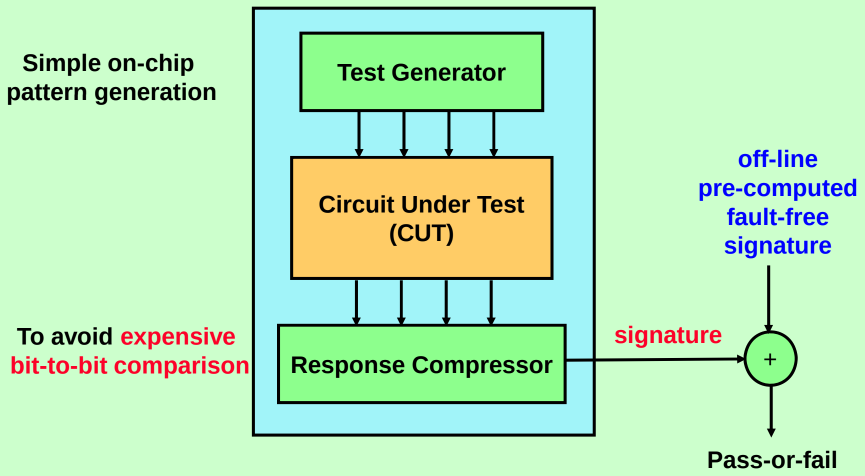
Can lead to significant test time reduction
Especially attractive for embedded cores

Good Things About BIST

- At-Speed Testing
  - catching timing defects
- Fast
  - reduce the testing time and testing costs
  - a major advantage over scan
- Board-level or system-level testing
  - can be conducted easily in field

ch7-5

General Organization of BIST



ch7-6

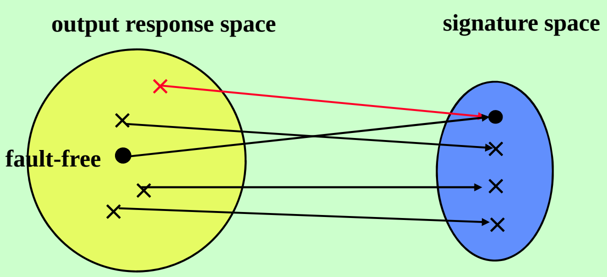
Why Compression ?

- Motivation
  - Bit-to-bit comparison is infeasible for BIST
- Signature analysis
  - Compress a very long output sequence into a single signature
  - Compare the compressed word with the pre-stored golden signature to determine the correctness of the circuit
- Problems
  - Many output sequences may have the same signature after the compression leading to the aliasing problem
  - Poor diagnosis resolution after compression

ch7-7

Aliasing Effect in Response Compression

- Aliasing - the probability that a faulty response is mapped to the same signature as the fault-free circuit (魚目混珠) 錯變成對的機率



Response compression is a mapping from the output response space to the signature space
In this example, aliasing prob. = 1 / 4 = 25%

ch7-8

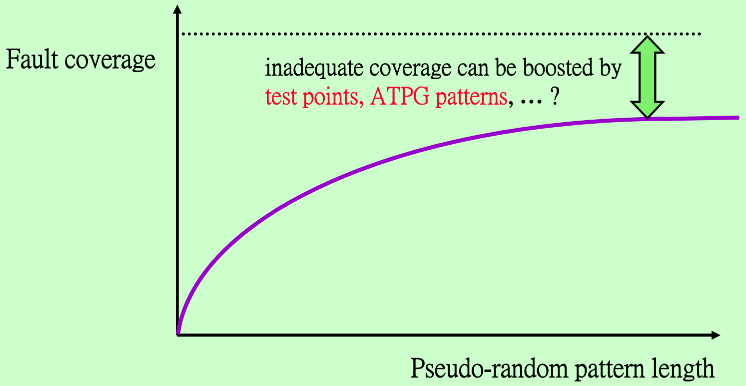
BIST Issues

- Area Overhead
- Performance Degradation
- Fault Coverage
  - Most on-chip generated patterns may not achieve a very high fault coverage
- Diagnosability
  - The chip is even harder to diagnose due to response compression

ch7-9

Random Pattern Resistant Faults

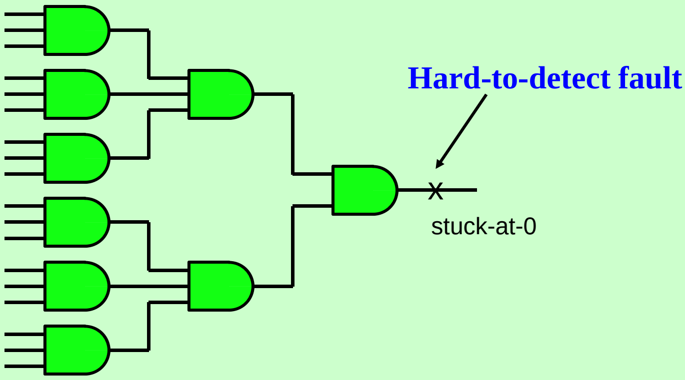
- An RPRF cannot be detected by random patterns
- is a major cause of low fault coverage in BIST



ch7-10

Example: Hard-To-Detect Fault

- Hard-to-detect faults
  - Faults that are not covered by random testing
  - E.g., an output signal of an 18-input AND gate



ch7-11

Reality of Logic BIST

- BIST is NOT a replacement for scan
  - it is built on top of full-scan
- BIST does NOT result in fewer patterns
  - it usually uses many more patterns than ATPG patterns
- BIST does NOT remove the need for testers
  - tester still required to
    - initiate test
    - read response
    - apply ATPG patterns to other part of IC

ch7-12

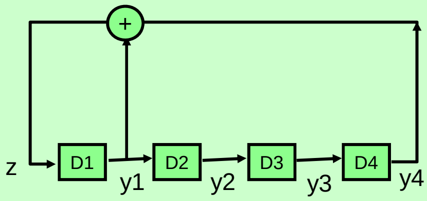
BIST Techniques

- Stored-Vector Based
  - Micro-instruction support
  - Stored in ROM
- Hardware-Based Pattern Generators
  - Counters
  - Linear Feedback Shift Registers
  - Cellular Automata

Linear Feedback Shift Register (LFSR)

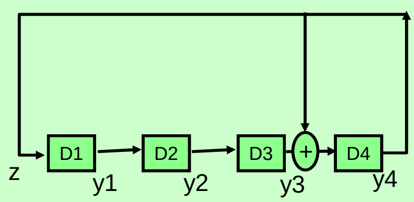
- Flip-Flop: one cycle delay
- XOR gate: modulo-2 addition
- Connection: modulo-2 multiplication

Type 1: Out-Tap



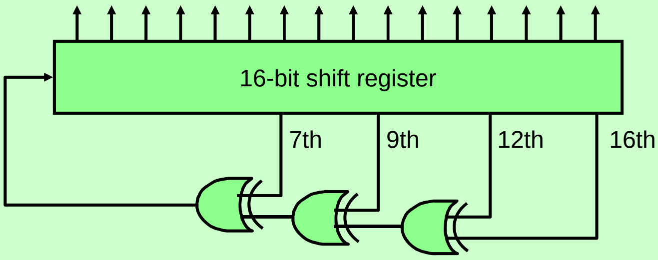
$z = y4 + y1 = D^4(z) + D(z)$

Type 2: In-Tap



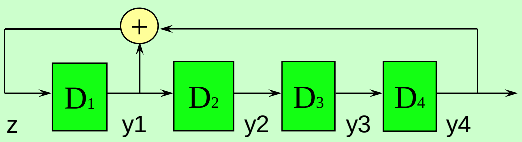
$z = y4 = D(y3 + y4) = D(D^3(z) + z)$
$= D^4(z) + D(z)$

LFSR – Example



This sixteen-stage LFSR will autonomously generates a maximum length of 2^{16}-1 = 65,535 state before the sequence repeats
The seed (I.e., initial state of the LFSR) should not be all-0 state.
All 0-state is called a forbidden seed.

LFSR Example



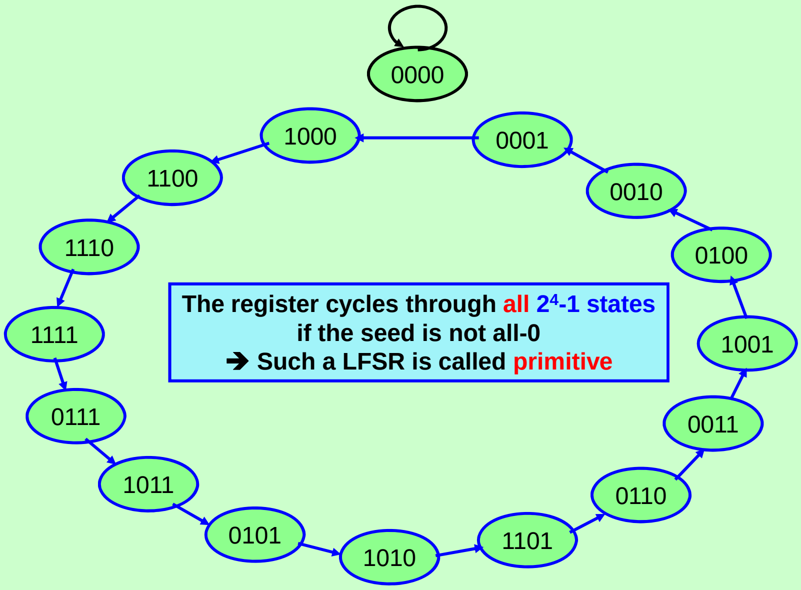
$$\begin{bmatrix} y1(t+1) \\ y2(t+1) \\ y3(t+1) \\ y4(t+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} y1(t) \\ y2(t) \\ y3(t) \\ y4(t) \end{bmatrix}$$

Characteristic polynomial

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

	D_4	D_3	D_2	D_1
	1	0	0	0
	0	0	0	1
	0	0	1	1
	0	1	1	1
	1	1	1	1
	1	1	1	0
	1	1	0	1
	1	0	1	0
	0	1	0	1
	1	0	1	1
	0	1	1	0
	1	1	0	0
	1	0	0	1
	0	0	1	0
	0	1	0	0
repeating ⟶	1	0	0	0

Ex: Primitive LFSR – State Diagram



The register cycles through all 2^4-1 states if the seed is not all-0
➔ Such a LFSR is called primitive

Primitive Polynomials (Up to Degree 100)

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

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

Galois Field GF(2)

- Operation
  - Modulo-2 addition, subtraction, multiplication, and division of binary data
- Properties
  - Modulo-2 addition and subtraction are identical
  - 0+0=0, 0+1=1, 1+0=1, 1+1=0
  - 0-0=0, 0-1=1, 1-0=1, 1-1=0

$(x^3 + x^2 + x + 1) \times (x^2 + x + 1)$ is given by:

$$
\begin{array}{r}
x^3 + x^2 + x + 1 \\
\underline{x^2 + x + 1} \\
x^3 + x^2 + x + 1 \\
x^4 + x^3 + x^2 + x \\
\underline{x^5 + x^4 + x^3 + x^2 } \\
x^5 + 0 + x^3 + x^2 + 0 + 1
\end{array}
$$

Bit-stream multiplication

Bit-stream division

$$
\begin{array}{r}
x^3 + x^2 + x + 1 \\
x^2 + x^1 + 1 \,\big)\, x^5 + 0 + x^3 + x^2 + 0 + 1 \\
\underline{x^5 + x^4 + x^3} \\
x^4 + 0 + x^2 \\
\underline{x^4 + x^3 + x^2} \\
x^3 + 0 + 0 \\
\underline{x^3 + x^2 + x} \\
x^2 + x + 1 \\
\underline{x^2 + x + 1} \\
0
\end{array}
$$

9

Why LFSR ?

- Simple and regular structure
  - D-flip-flops and XOR gates
- Compatible with scan DFT design
- Capable of exhaustive and/or pseudo exhaustive testing
  - If the LFSR is properly configured
- Low aliasing probability
  - The fault coverage lost due to the response compression is less than other compression schemes

ch7-20

LFSR – Definitions

- Maximum-length sequence
  - A sequence generated by an n-stage LFSR is called a maximum-length sequence if it has a period of 2^n-1
  - A maximum-length sequence is called m-sequence
- Primitive polynomial
  - The characteristic polynomial associated with a maximum-length sequence is called a primitive polynomial
- Irreducible polynomial
  - A polynomial is irreducible if it cannot be factorized into two (or more) parts, I.e., it is not divisible by any polynomial other than 1 and itself.

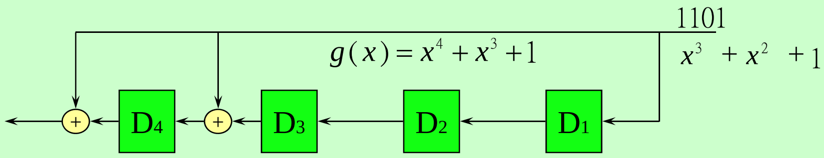
ch7-21

LFSR – Properties

- No. of 1s and 0s
  - The number of 1s in an *m*-sequence differs from the number of 0s by only one
- Pseudo-random sequence
  - The sequence generated by an LFSR is called a pseudo-random sequence
- The correlation
  - Between any two output bits is very close to zero
- Consecutive run of 1s and 0s
  - An *m*-sequence produces an equal number of runs of 1s and 0s.
  - In every *m*-sequence, one half the runs have length 1, one fourth have length 2, one eighth have length 3, and so forth

ch7-22

LFSR – Polynomial Multiplication



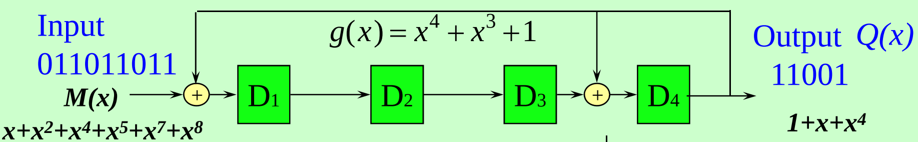
Add-and-shift

Output stream	$D_4 D_3 D_2 D_1$	Input stream
	0 0 0 0	1 1 0 1
1	1 0 0 1	1 0 1
1 0	1 0 1 1	0 1
1 0 1	0 1 1 0	1
1 0 1 1	0 1 0 1	
x^7 x^5 x^4	x^2 1	

$$\left(x^4 + x^3 + 1\right) \times \left(x^3 + x^2 + 1\right) = x^7 + x^5 + x^4 + x^2 + 1$$

ch7-23

LFSR – Polynomial Division (Example)



$M(x)$	$D_1 D_2 D_3 D_4$	$Q(x)$ Quotient
0 1 1 0 1 1 0 1 1	0 0 0 0	
after 4 shifts → 0 1 1 0 1	1 0 1 1	
0 1 1 0	0 1 0 0	1
0 1 1	0 0 1 0	0 1
0 1	1 0 0 1	0 0 1
0	0 1 0 1	1 0 0 1
	1 0 1 1	1 1 0 0 1
Remainder →	1 $+x^2+x^3$	1 $+x$ $+x^4$

$(x^8+x^7+x^5+x^4+x^2+x) \div (x^4+x^3+1) = x^4+x+1$

$R(x) = x^3+x^2+1$

ch7-24

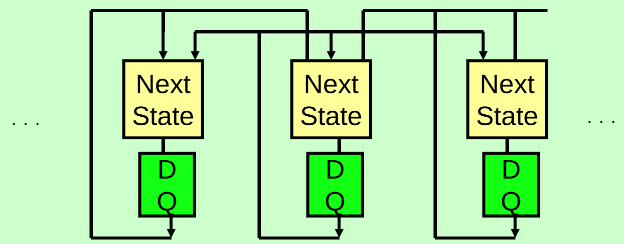
LFSR – Summary

- LFSRs have two types
  - In-tap and Out-tap
- LFSRs
  - Can be used to implement polynomial multiplication and division in GF(2)
- As polynomial multiplier
  - LFSRs are capable of generating pseudo random vectors
- As polynomial divisors
  - LFSRs are capable of compressing test response

ch7-25

Cellular Automaton (CA)

- An one-dimensional array of cells
- Each cell contains a storage device and next state logic
- Next state is a function of current state of the cell and its neighboring cells



Three-cell neighbor

ch7-26

Cellular Automata – Name

- Name of CA functions
  - Is determined by its truth table

State	A0	A1	A2	A3	A4	A5	A6	A7
C_{i+1}	0	0	0	0	1	1	1	1
C_i	0	0	1	1	0	0	1	1
C_{i-1}	0	1	0	1	0	1	0	1

Next State K-Map F_{CA}

A0	A2	A4	A6
A1	A3	A5	A7

$$Name = \sum_{i=0}^{7} A_i 2^i$$ **(defined by Wolfram)**

Example: $F_{CA} = C_{i-1} \oplus C_i$

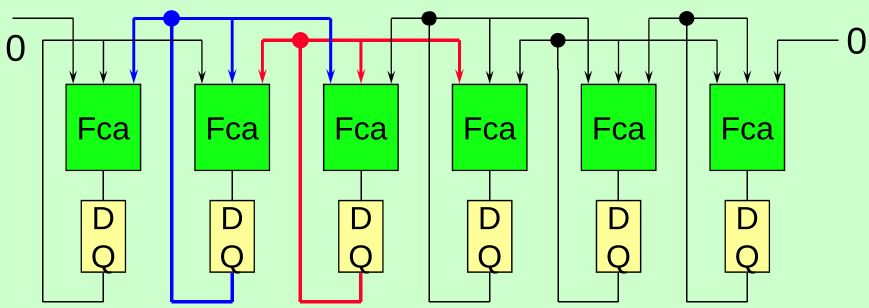
C_{i+1} \ $C_i C_{i-1}$	00	01	11	10
0	**0**	**1**	**0**	**1**
1	**0**	**1**	**0**	**1**

Name = 64+32+4+2
= 102

ch7-27

Cellular Automata – Hardware

CA with Null Boundary Condition

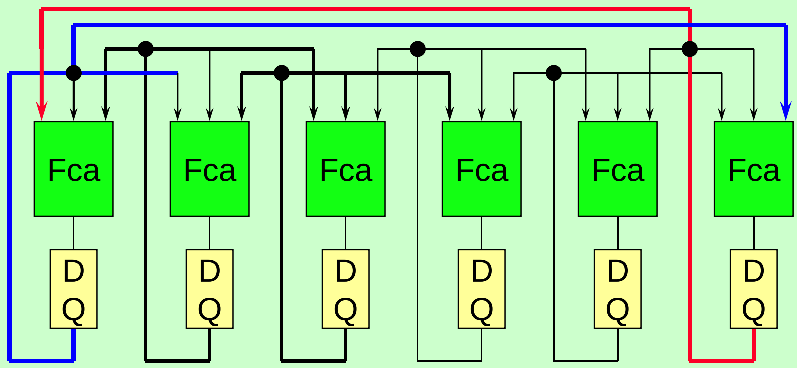


Standard – All the CAs are of the same type
Hybrid – The CAs are of different type

ch7-28

Cellular Automata – Hardware

CA with cyclic Boundary Condition



Outline

- Basics
- Test Pattern Generation
  - How to generate patterns on chip using minimum hardware, while achieving high fault coverage
- Response Analyzers
- BIST Examples
- Memory BIST

On-Chip Pattern Generation

PG Hardware	Pattern Generated
• Stored Patterns	• Deterministic
• Counter Based	• Pseudo-Exhaustive
• LFSR Based	• Pseudo-Random
• Cellular Automata	• Pseudo-Random

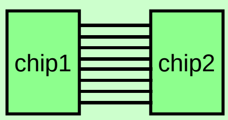
Pseudo Random Patterns: Random patterns with a specific sequence defined by a seed

ch7-31

Counter Based Pattern Generation

- Generates regular test sequences
  - Such as walking sequence and counting sequence for memory interconnect testing

cycle	Walking Sequence	Counting Sequence
1	1 0 0 0 0 0 0 0	0 0 0
2	0 1 0 0 0 0 0 0	0 0 1
3	0 0 1 0 0 0 0 0	0 1 0
4	0 0 0 1 0 0 0 0	0 1 1
5	0 0 0 0 1 0 0 0	1 0 0
6	0 0 0 0 0 1 0 0	1 0 1
7	0 0 0 0 0 0 1 0	1 1 0
8	0 0 0 0 0 0 0 1	1 1 1
line id	1 2 3 4 5 6 7 8	

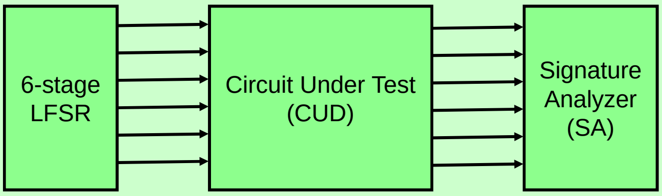


coupling between interconnects can be tested by walking sequence

ch7-32

On-Chip Exhaustive Testing

- Exhaustive testing
  - Apply all possible input combinations to CUD
  - A complete functional testing
  - 100% coverage on all possible faults
- Limitation
  - Only applicable for circuits with medium number of inputs



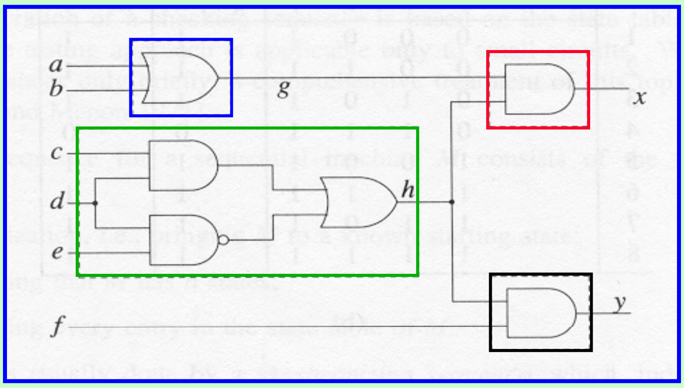
ch7-33

Pseudo Exhaustive Testing (PET)

- Apply all possible input combinations to every partitioned sub-circuits
- 100% fault coverage on single faults and multiple faults within the sub-circuits
- Test time is determined by the number of sub-circuits and the number of inputs to the sub-circuit
- Partitioning is a difficult task

ch7-34

Example for Pseudo-Exhaustive Testing



10 vectors are enough to pseudo-exhaustively test this circuit, Compared to 2^6=64 vectors for naive exhaustive testing

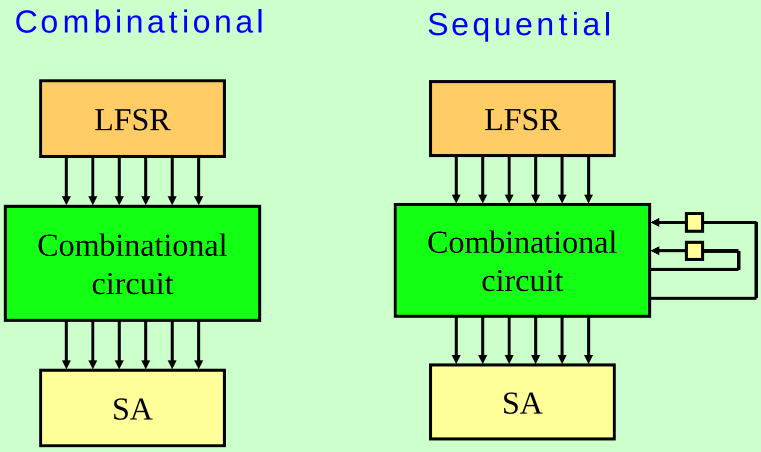
ch7-35

LFSR-Based Pattern Generation

- Apply random test sequence generated by LFSR/CA
- Simplest to design and implement
- Lowest in hardware overhead
- Fault coverage
  - Is a function of the test length and the random testability of the circuits
  - Certain circuits are more resistant to random patterns than others

ch7-36

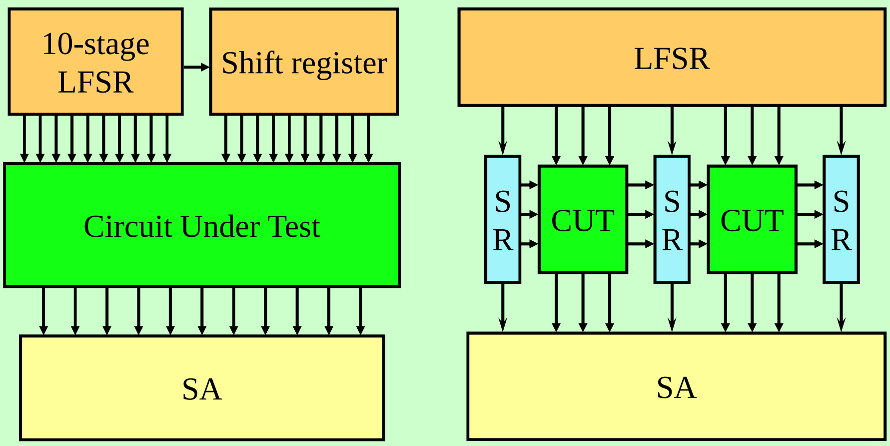
Pseudo Random Testing Hardware



(Circular BIST)

ch7-37

BIST – Pseudo Random Testing Hardware



(CEBT) (STUMPS)

test-per-clock configuration

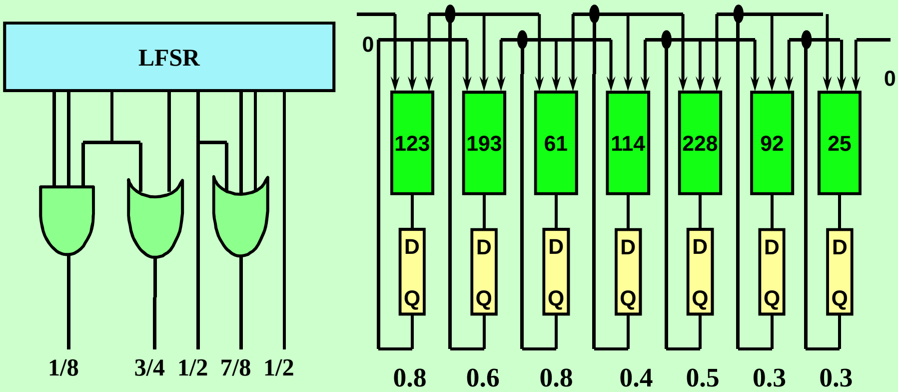
test-per-scan configuration

ch7-38

Weighted Pseudo Random Testing

It was observed that weighted random patterns could achieve higher fault coverage in most cases !

LFSR Based | **Weighted Cellular Automaton**



ch7-39

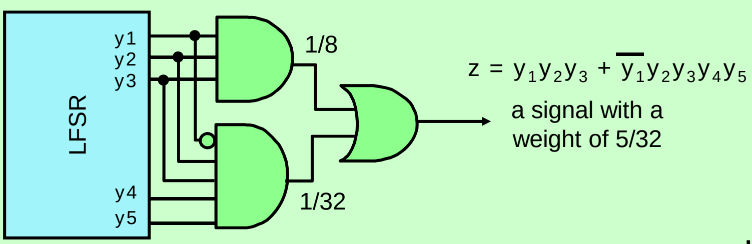
Signal of An Arbitrary Weight

- To implement a signal
  - with a signal-1 probability (weight) of 5/32
- Procedure

  (1) Decompose into a sum of basic weights

  5/32 = 4/32 + 1/32 = 1/8 + 1/32

  (2) Use AND and OR gates to realize the weight



ch7-40

Outline

- Basics
- Test Pattern Generation
- Response Analyzers
  - How to compress the output response without losing too much accuracy
- BIST Examples
- Memory BIST

ch7-41

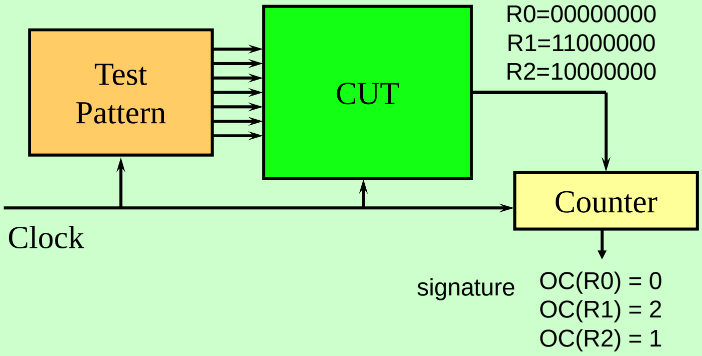
Types of Response Compression

- Ones-counting compression
- Transition-counting compression
- Signature Analysis

ch7-42

Ones-Counting Signature

- Procedure
  - Apply the predetermined patterns
  - Count the number of ones in the output sequence

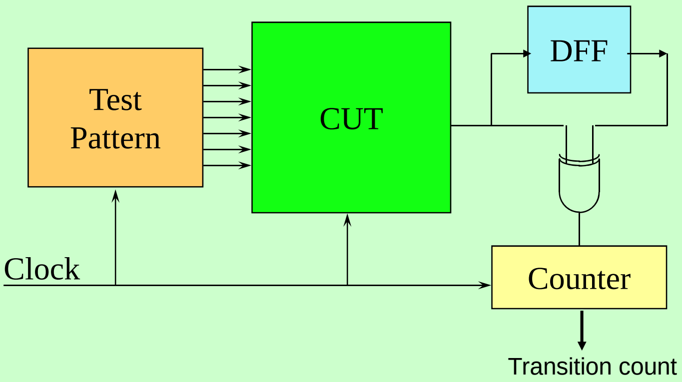


Zero-Aliasing Test Set for Ones-Counting

- Notations
  - T0: set of test vectors whose fault-free response is 0
  - T1: set of test vectors whose fault-free response is 1
- Theorem
  - The following new test set does NOT suffer from fault masking using ones count testing
  - T = {T0, (|T0|+1) copies of every pattern in T1}
  - Note that the fault masking only occurs when a fault is detected by the same number of patterns in T0 and T1; the above new test set avoid this condition

Transition-Counting Signature

- Procedure
  - Apply predetermined patterns
  - Count the number of 0→1 and 1→0 transitions



ch7-45

Aliasing of Transition-Counting

- Consider a sub-sequence of bits

  (… r_{j-1} r_j r_{j+1} …)

  If r_{j-1} is not equal to r_{j+1}, then an error occurring at r_j will not be detected by transition counting.

- Example

1. (0, 1, 1) → (0, 0, 1)
2. (0, 0, 1) → (0, 1, 1)
3. (1, 1, 0) → (1, 0, 0)
4. (1, 0, 0) → (1, 1, 0)

ch7-46

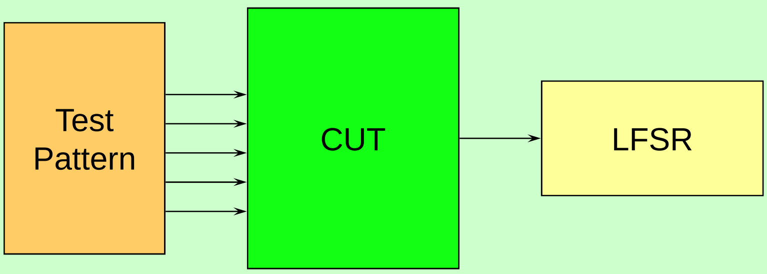
Aliasing of Transition Counting

- Aliasing Probability
  - Notations
    - m: the test length
    - r: the number of transitions
  - Highest when r=m/2
  - No aliasing when r=0 or r=m
  - For combinational circuits, permutation of the input sequence results in a different signature
  - One can reorder the test sequence to minimize the aliasing probability

ch7-47

Signature Analysis by LFSR

- Procedure
  - Apply predetermined patterns
  - Divide the output sequence by LFSR



ch7-48

Example: Aliasing Probability

- Assume that
  - Output number to be compressed has $m=4$ bits
  - The compression is done by dividing output number by a divisor of 2^n-1, (e.g., the divisor is $2^2-1 = 3$ when $n=2$)
  - The remainder is taken as the signature
- Possible signatures

output = 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

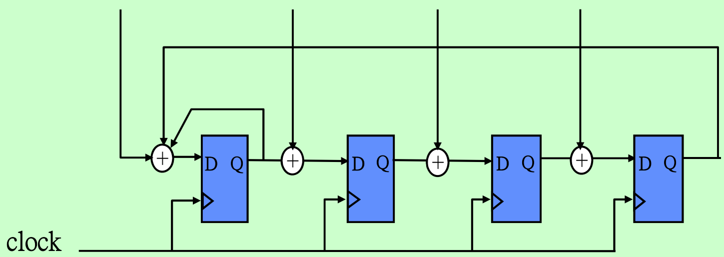
remainder = 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0

aliasing prob. when signature is 0 = $(2^m/(2^n-1)) / 2^m$

$= 1/(2^n-1) \sim 2^{-n}$

Multiple Input Shift Register (MISR) (Temporal Compression)

- A MISR compacts responses from multiple circuit outputs into a signature



Aliasing probability of m stage = 2^{-m}

Outline

- Basics
- Test Pattern Generation
- Response Analyzers
- BIST Examples
- Memory BIST

ch7-51

Key Elements in a BIST Scheme

- Test pattern generator (TPG)
- Output response analyzer (ORA)
  - Also called Signature Analyzer (SA)
- The circuit under test (CUT)
- A distribution system (DIST)
  - which transmits data from TPG's to CUT's and from CUT's to ORA's
  - e.g., wires, buses, multiplexers, and scan paths
- A BIST controller
  - for controlling the BIST circuitry during self-test
  - could be off-chip

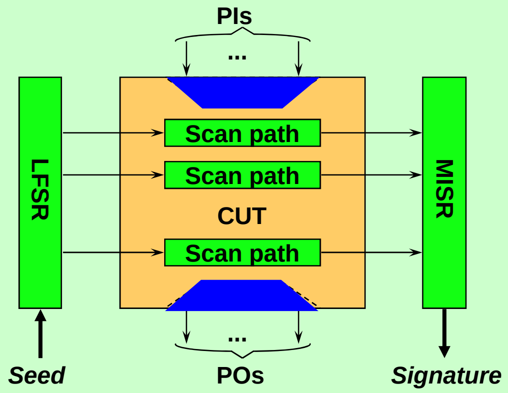
ch7-52

HP Focus Chip (Stored Pattern)

- Chip Summary
  - 450,000 NMOS devices, 300,000 Nodes
  - 24MHz clocks, 300K-bit on-chip ROM
  - Used in HP9000-500 Computer
- BIST Micro-program
  - Use microinstructions dedicated for testing
  - 100K-bit BIST micro-program in CPU ROM
  - Executes 20 million clock cycles
  - Greater than 95% stuck-at coverage
  - A power-up test used in wafer test, system test, field test

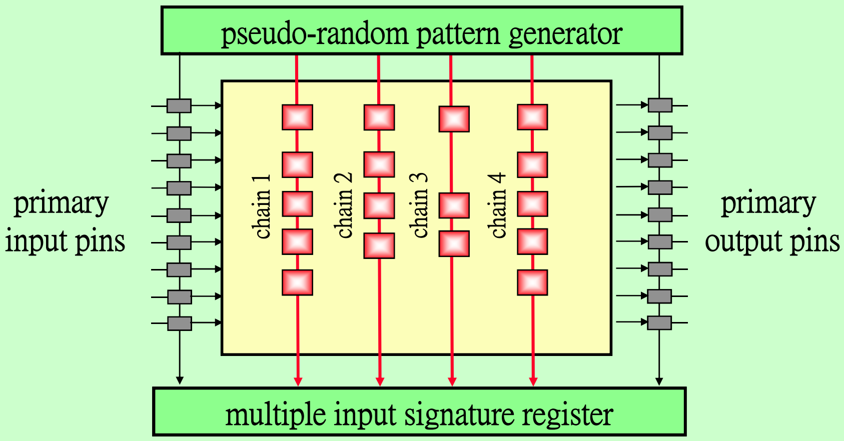
Logic BIST Example

- Features
  - [Bardell 1982, 84]
  - Self-Test using LFSR and Parallel MISR
  - Multiple scan chains to reduce test time

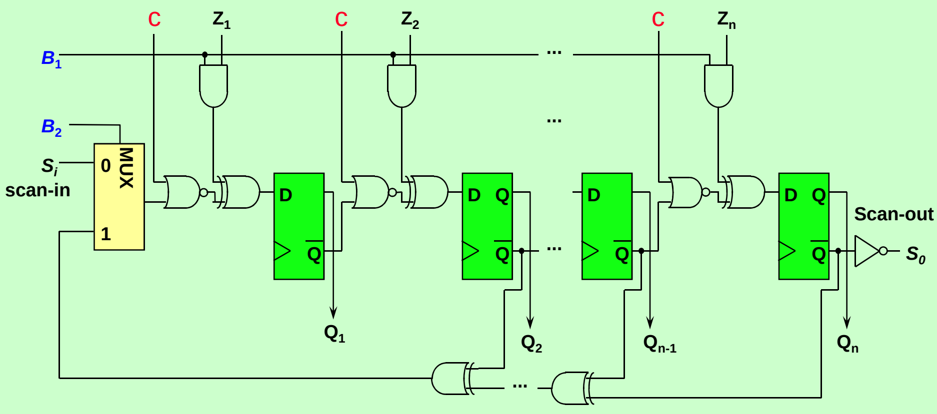


Scan-Based Logic BIST Architecture

called STUMPS architecture by Mentor Graphics



Built-In Logic Block Observation (BILBO)

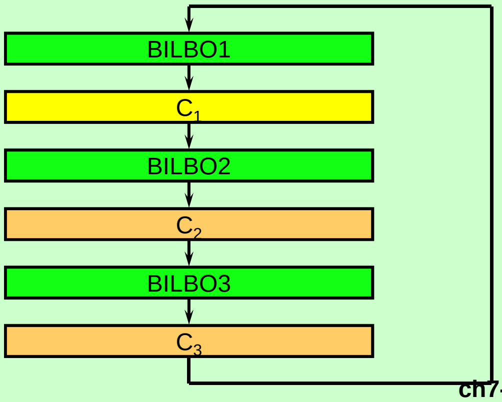


B_1	B_2	operation mode	C
0	0	shift register	0
0	1	LFSR pattern generation	0
1	1	MISR response compressor	0
1	0	parallel load (normal operation)	1

Example: BILBO-Based BIST

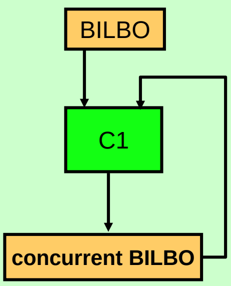
- Test procedure
  - each logic block C1, C2, C3 are tested in a serial manner
  - BIST controller needs to configure each BILBO registers properly during self-testing

when testing C1
BILBO1 is a PRPG
BILBO2 is a MISR



Concurrent BILBO

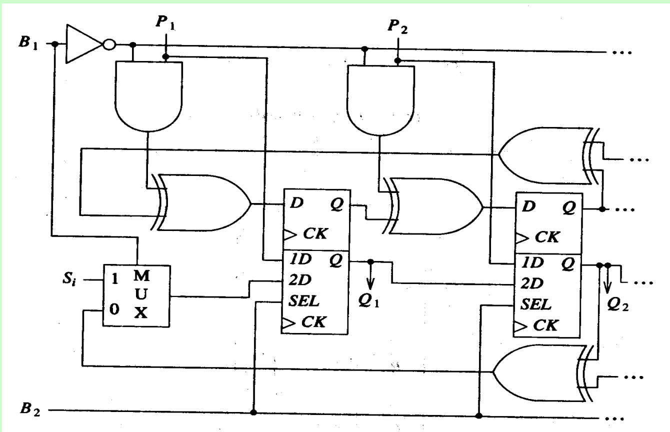
Logic with self-loop



needs to be PRPG and MISR simultaneously

B_1	B_2	Operation mode
—	0	Normal
1	1	Scan
0	1	PRPG/MISR

top-row of D-FFs → MISR
bottom-row of D-FFs → PRPG



Outline

- Basics
- Test Pattern Generation
- Response Analyzers
- BIST Examples
- Memory BIST

ch7-59

The Density Issues

- Historical π-Rule
  - The number of bits per chip has quadrupled roughly every 3.1 (or π) years
- Density Induced Faults
  - The cells are closer together
  - More sensitive to influences of neighbors
  - More vulnerable to noise on the address and data lines

ch7-60

Test Time May Get Too Long !

- For today's memory chips
  - Test time becomes a big issue !
  - We can afford nothing but linear test algorithm
- Example
  - assume that the clock cycle time is 100 ns

Algorithm complexity / Capacity n	Testing time (in seconds)			
	64n	n•$\log_2$n	$3n^{3/2}$	$2n^2$
16k	0.1	0.023	0.63	54
64k	0.4	0.1	5.03	14 Mins
256k	1.7	0.47	40.3	3.8 Hrs
1M	6.7	2.1	5.4 Mins	61 Hrs
4M	26.8	9.2	43 Mins	41 Days
16M	1.8 Mins	40.3	5.7 Hrs	2 Years

ch7-61

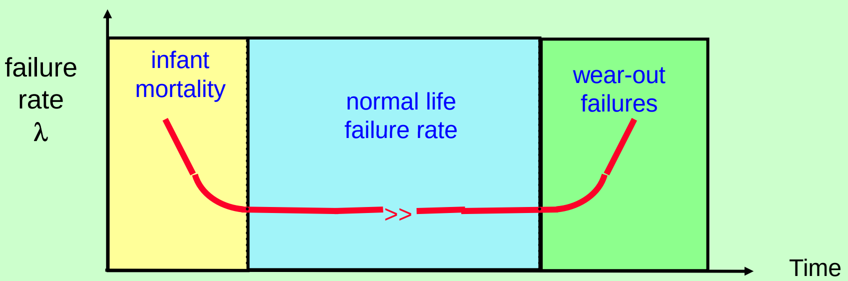
IC Failure Rate Versus Time

Def: failure rate

The no. of failures per unit time as a fraction of total population

IC's failure rate is like a bathtub curve with three stages:

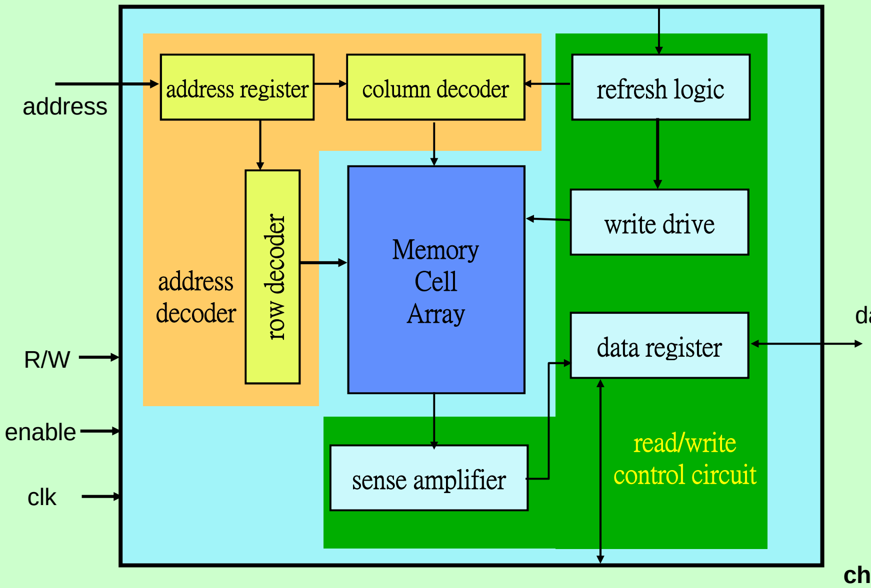
1. Infant mortality stage: typically a few weeks
2. Normal life failure stage: up to 25 years or so
3. Wear-out stage



Short period of accelerated stress test prior to shipment
→ To eliminate the infant mortality

ch7-62

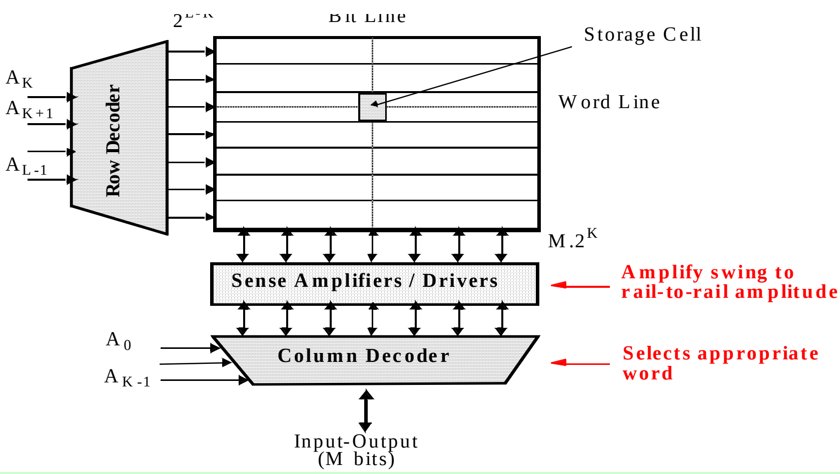
Memory Model



ch7-63

Memory Array

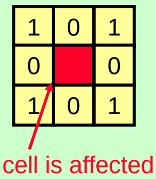
Problem: ASPECT RATIO or HEIGHT >> WIDTH



ch7-64

Fault Models

- Stuck-At Faults (SAF)
  - cell, data line, address line, etc.
- Open Faults (SAF)
  - open in data line or in address line
- Transition Faults (TF)
  - Cell can be set to 0, but not to 1
- Address Faults (AF)
  - faults on decoders
- Coupling Faults (CF)
  - short or cross-talk between data (or address) lines
  - A cell is affected by one of its neighboring cells
- Neighborhood Pattern Sensitive Fault (NPSF)
  - A cell is affected by when its neighbors form a pattern



ch7-65

Example Faults

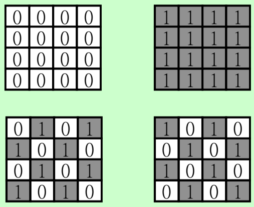
Fault Models

- SAF : Cell stuck
- SAF : Driver stuck
- SAF : Read/write line stuck
- SAF : Chip-select line stuck
- SAF : Data line stuck
- SAF : Open in data line
- CF : Short between data lines
- CF : Cross-talk between data lines
- AF : Address line stuck
- AF : Open in address line
- AF : Open decoder
- AF : Shorts between address lines
- AF : Wrong access
- AF : Multiple access
- TF : Cell can be set to 0 but not to 1 (or vice-versa)
- NPSF : Pattern sensitive interaction between cells

ch7-66

Simple Test Algorithms

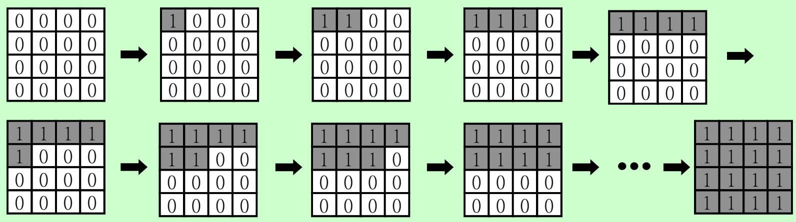
- Test Algorithm
  - is an abstract description of a sequence of test patterns.
- Commonly Used Algorithms
  - Background patterns
  - Checkerboard patterns
  - March Patterns



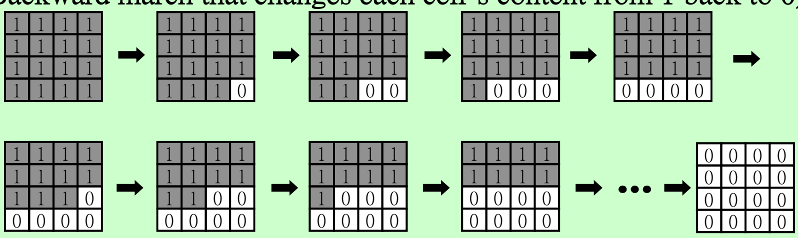
ch7-67

A March Algorithm

(Forward march that changes each cell's content from 0 to 1)

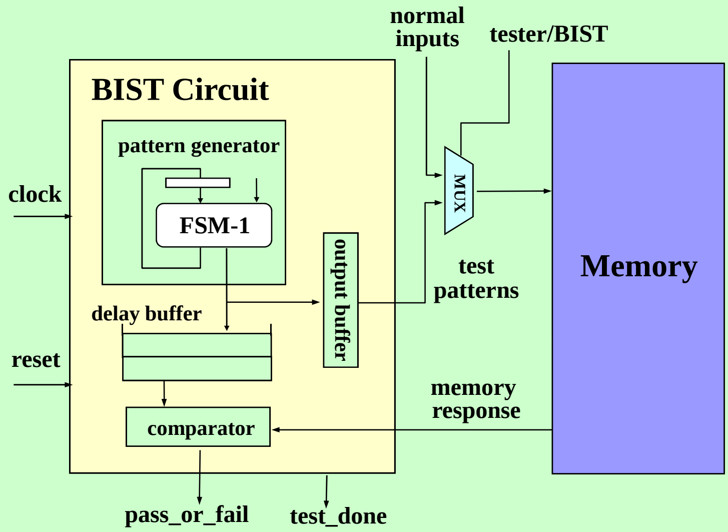


(Backward march that changes each cell's content from 1 back to 0)



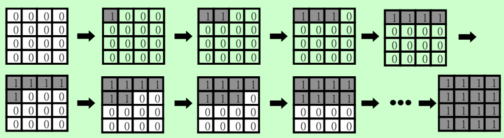
ch7-68

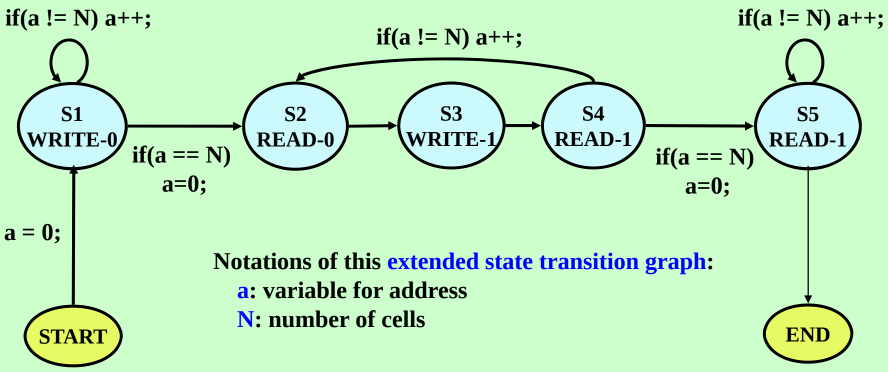
Example: A Memory BIST



ch7-69

Finite State Machine for March Alg.

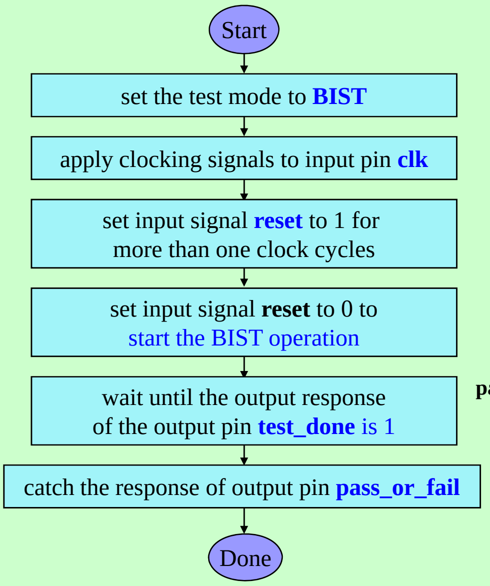


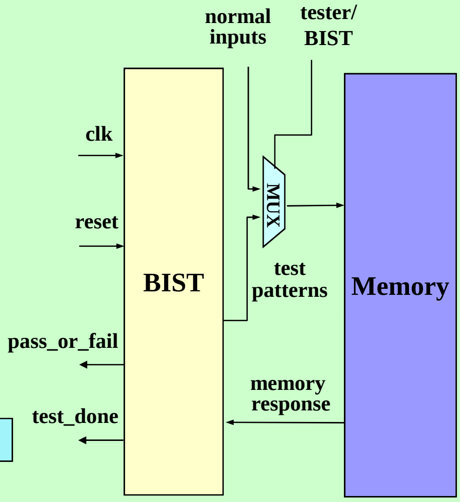


Notations of this extended state transition graph:
- **a: variable for address**
- **N: number of cells**

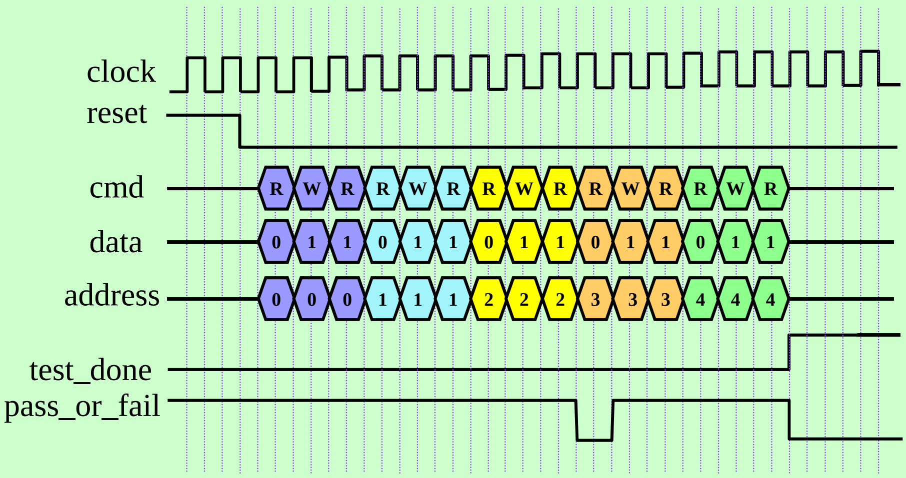
ch7-70

Testing Procedure of BISTed Memory





A Waveform Example



Quality Measures of BIST

BIST-vs.-Tester Profile		Tester	
		pass	fail
B I S T	pass	(I)	(III) 漏網之魚
	fail	(II) 誤殺者	(IV)

To minimize region (II) and (III):

1. False Negative Ratio: (II) / #chips e.g., (1/20) = 5%
2. False Positive Ratio: (III) / #chips e.g., (2/20) = 10%