Built In Self Test (BIST)

Pattern Generation using BIST Engine

2. **Pseudo Random BIST**

When pseudo-random testing is performed on a memory, the outputs cannot be compared with expected data. Since the data was written into the memory in a pseudo-random fashion, the output is not very predictable.

Another pseudo-random technique is utilized for compressing the output results and determining whether the resulting test passed or failed.

A multiple input signature register or MISR can take the data output values read from the memory during the entire test sequence and generate a string of bits that indicate a pass/fail condition.

Reference :“High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

ATE and BIST

Automated Test Equipment (ATE):

- If there is a **memory tester available it doesn't make sense to use BIST on a stand-alone memory,** since the IO are available.
- The large ATE memory tester is far more powerful than a BIST and therefore should be **utilized for stand-alone memories** but not for embedded ones.

BIST

- BIST is highly **optimized for embedded memories** as Embedded Memories does not have IOs.
- BIST makes sense for stand-alone memories as well. When a stand-alone memory is used in a multichip module (MCM), **the tester for** the **MCM is a logic tester.**
- Having BIST on the memory chip enables test of the memory in this logic environment, which would otherwise be impossible or at least vastly inefficient and of limited quality.
- Thus BIST can be included on the stand-alone memory to facilitate its test during MCM manufacturing.

Memory may be

- ✓ Stand Alone Memory
- ✓ Embedded Memory

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

PES UNIVERSITY ONLINE

Memory Design and Testing

Mahesh Awati

Department of Electronics and Communication Engg.

MEMORY DESIGN AND TESTING

UNIT 5 – Memory Self Test

Mahesh Awati

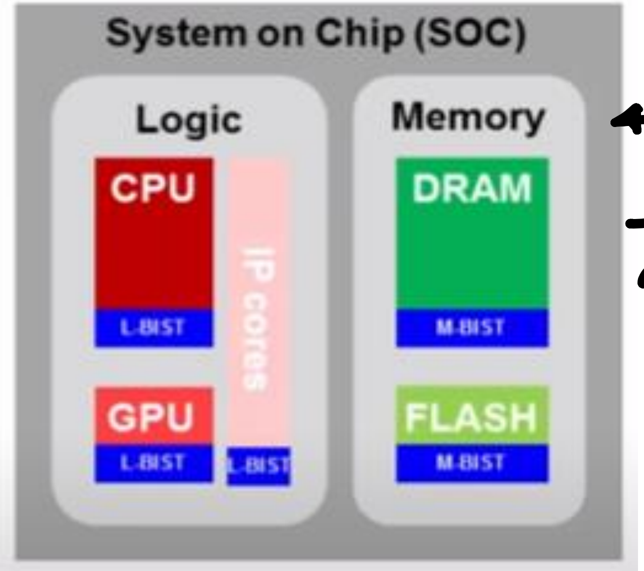
Department of Electronics and Communication Engineering

Built In Self Test (BIST)

Introduction

Basic Concept

- **Efficient, thorough and easy way of testing complex embedded memory** without any connection to the outside world
- In this, **an extra hardware is added to the chip for test generation** and response evaluation.
- **Test is developed concurrently with the design s**ince the BIST needs to be designed onto the chip
  - ✓ Done on chip.
  - ✓ Additional hardware overhead.
- Only a few external pins to control BIST operation.
  - **Input pin: TEST CONTROL (TC)**
  - **Output pin: GOOD/BAD**
- Understanding **of both design and test is required to develop a good BIST engine**
- **Addition of BIST to memory should not degrade memory performance**



TC ←

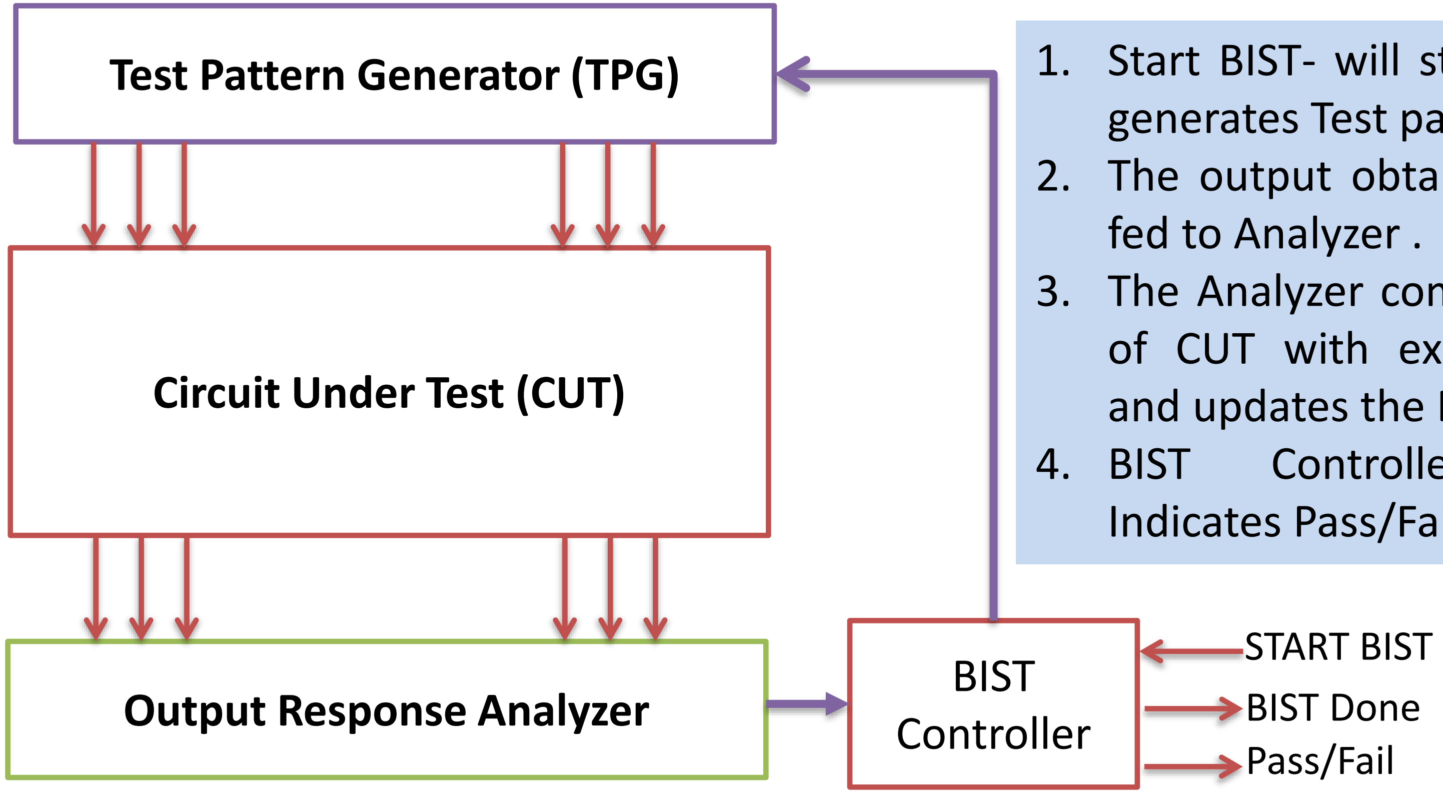
→ Good /bad

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

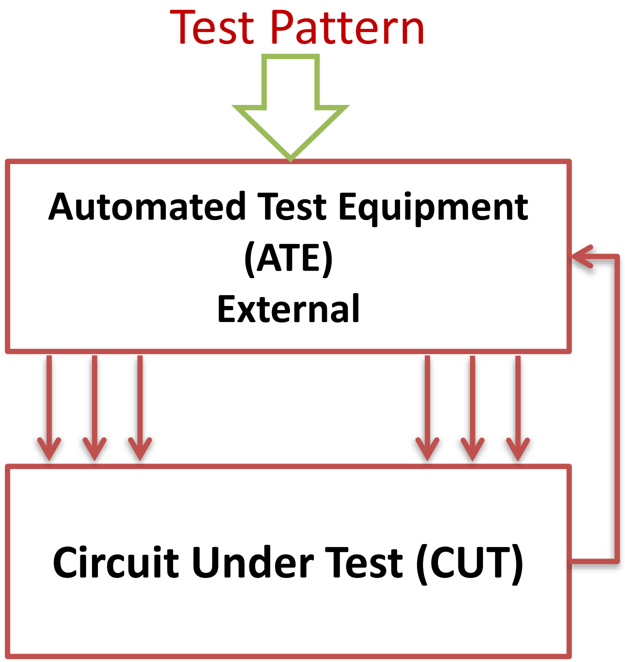
Built In Self Test (BIST)

Introduction

Basic Architecture of BIST



1. Start BIST- will start TPG which generates Test patterns .
2. The output obtained by CUT is fed to Analyzer .
3. The Analyzer compares the o/p of CUT with expected output and updates the BIST Controller.
4. BIST Controller generator Indicates Pass/Fail

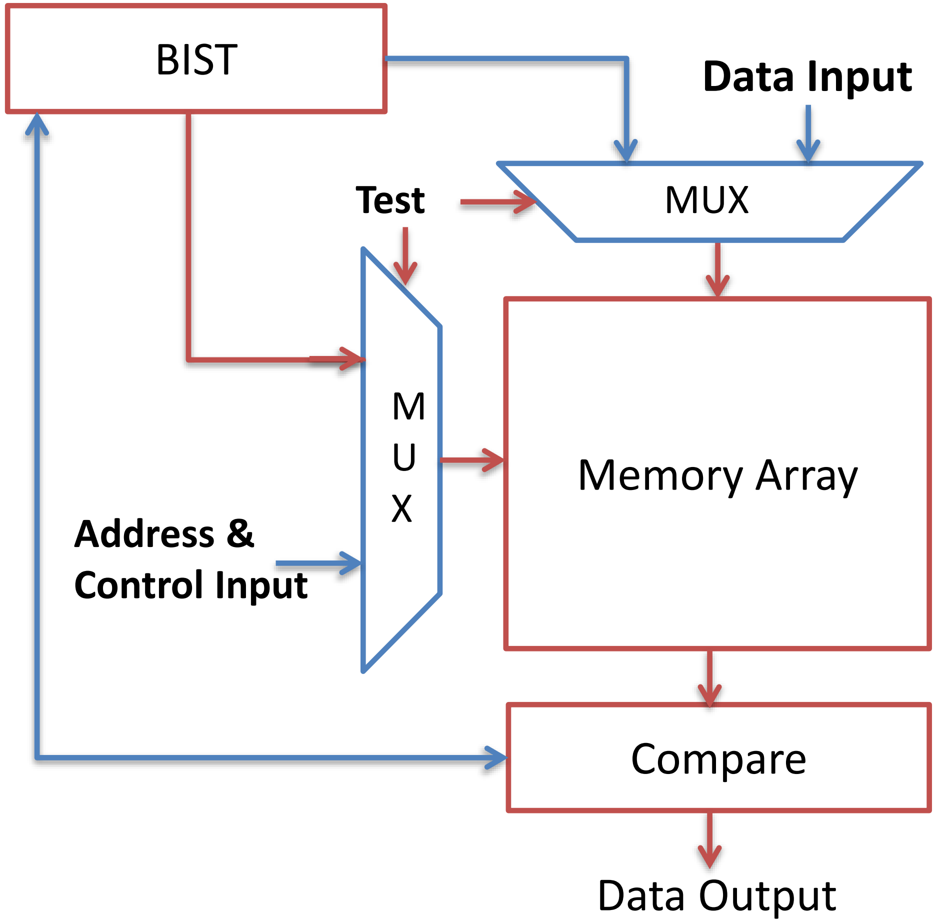


Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)
The Memory Boundary

THE MEMORY and Memory BIST Interaction

- **Boundary through which BIST accesses memory should be clean**. In other words there should be a **minimum of logic in between the boundary and the memory array** itself.
- In this manner the BIST engine can apply the needed patterns without having to concern itself with conditioning preceding logic (Minimum logic inside).
- Minimum logic between the memory array and the output compare circuit.
- The inputs to the MUX are from functional path and BIST engine. During Test, the BIST path is selected
- **Output from the memory is compared with the correct output stored in BIST.** A pass/fail indication is sent back to the BIST engine.
- MUX arrangement is used on address and control inputs as well



Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

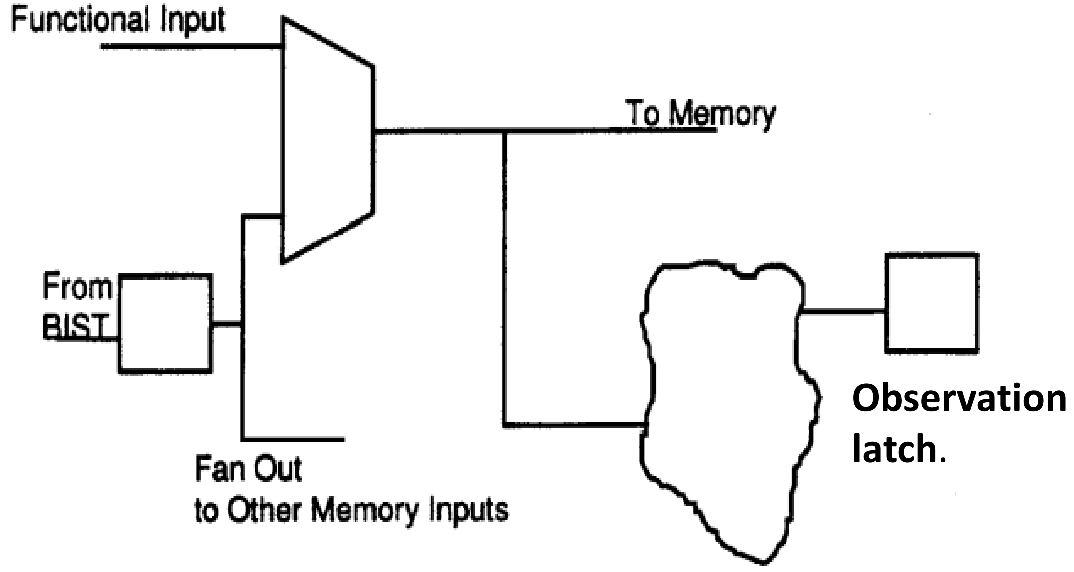
The Memory Boundary

THE MEMORY and Memory BIST Interaction (Logic Test)

- There are Logic circuit and Memory Array Involved. Both Logic circuit as well as Memory Array need to be tested.
- The MUX is involved in selecting the Functional Input and BIST engine output.

What if the MUX is Faulty?

- A slightly more complex arrangement **is one where the output of the multiplexer is routed to an observation latch**.
- In this manner, the logic test boundary continues beyond the multiplexer output into the observation latch.
- The multiplexer can, in this way, be fully tested.
- **During logic test**, signals are **sent from the BIST engine** and **captured into the latch**.
- Furthermore, during logic test **signals are sent along the functional path** and captured into the multiplexer. Thus the IO of the multiplexer are fully tested during logic test.



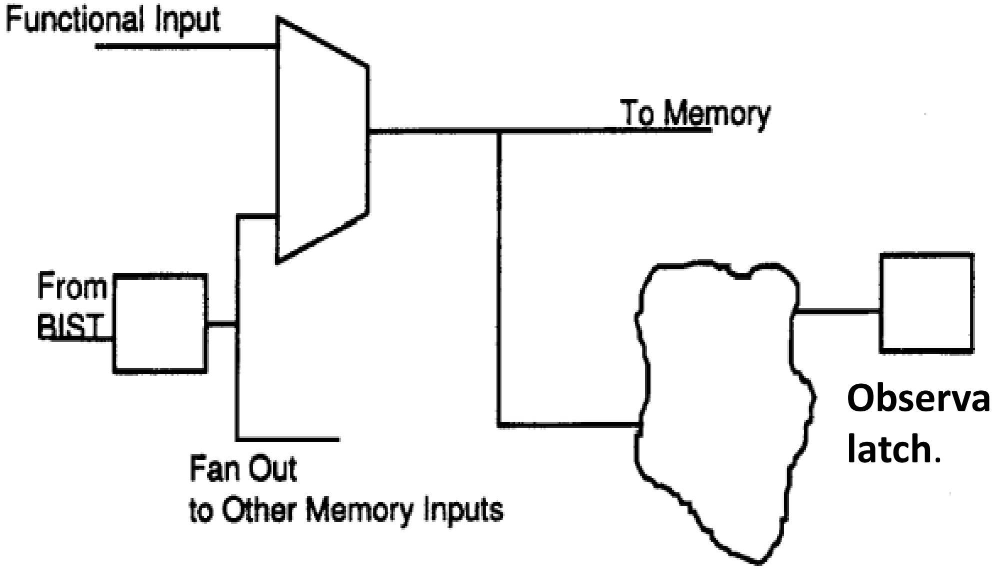
- The output of the multiplexer can be observed, either directly or after some combinational logic, illustrated by the bubble

Reference :“High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)
The Memory Boundary

THE MEMORY and Memory BIST Interaction (Logic Test)

- The input from BIST can go directly to the multiplexer or it can be used as shown here i.e., the signal from BIST is latched and can be fanned out to other memory inputs along with MUX.
- This configuration can reduce the number of BIST to memory signals and improve the speed at which the memory/BIST combination can run.
- Almost all embedded memories are Synchronous memories, which require a clock input.
- **The clock that exercises the memory during BIST can be**
  - ✓ the **same clock as that used during functional operation;** this method is generally preferred.
  - ✓ **Alternatively, there can be a second clock supplied to the memory during BIST test** which goes through a multiplexer-type arrangement.



Observation latch.

Reference :“High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

ATE and BIST

Automated Test Equipment (ATE):

- If there is a **memory tester available it doesn't make sense to use BIST on a stand-alone memory,** since the IO are available.
- The large ATE memory tester is far more powerful than a BIST and therefore should be **utilized for stand-alone memories** but not for embedded ones.

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

Pattern Generation using BIST Engine

The BIST used for Memory devices can be classified based on type of Pattern they generate for Memory testing.

1. **Deterministic Pattern Generating BIST**

- **Patterns generated follow specific pre-determined values**
- Pattern values are defined by the logic of the BIST design
- BIST generates algorithmic patterns

Ex: March C- patterns ; accesses each cell 14 times. The operations performed in the second element are a read "0" followed by a write "1" on each addresses.

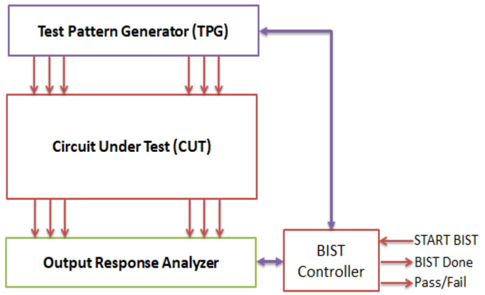
- Thus this specific, regular pattern is deterministic.

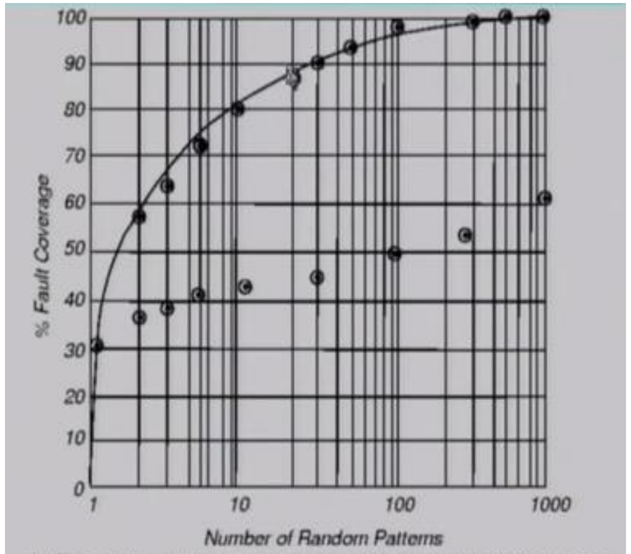
2. **Random Pattern Generating BIST :** The opposite of deterministic are patterns that are random.

i. The Faults coverage by Random Pattern generated depends on the Circuit Under Test which may

  - **Have Acceptance to Random Pattern**
  - **Be Resistance to Random Pattern**

ii. The Comparison : If the pattern is purely Random then, how to compare the response of Circuit Under Test because the expected output is not known for random pattern.





(a) Top curve -- random pattern testing with acceptable fault coverage.
(b) Bottom curve -- unacceptable random pattern testing.

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Pattern Generation using BIST Engine

Pseudo Random BIST

- **No BIST patterns are truly random.**
- Even those patterns which are pseudo- **random are not random but follow specific sequence defined by the logic.** Thus pseudo-random patterns are deterministic.
- A pseudo-random pattern is generated by a **Linear Feedback Shift Register or LFSR**. Another name for a LFSR is a **Pseudo-Random Pattern Generator or PRPG**

- **The Standard LFSR**
  - ✓ Produce patterns algorithmically – **repeatable**
  - ✓ Has most **desirable random properties**
  - ✓ **Need not to cover** all 2^n input combinations
  - ✓ **Long sequence are needed** for **good Fault coverage**

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

Pattern Generation using BIST Engine

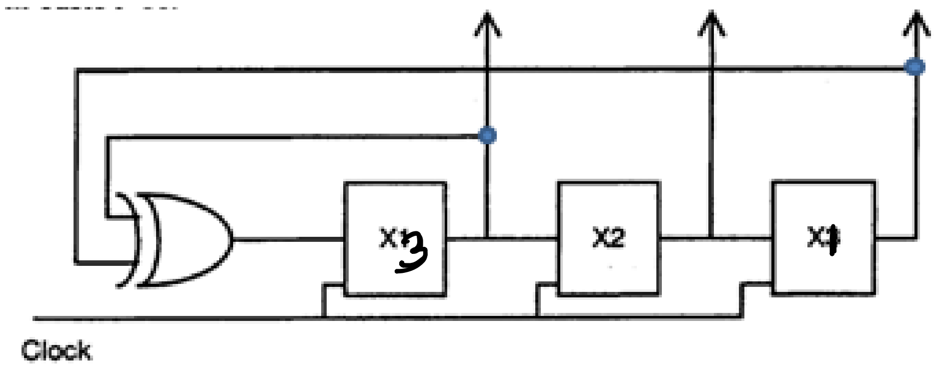
Pseudo Random BIST

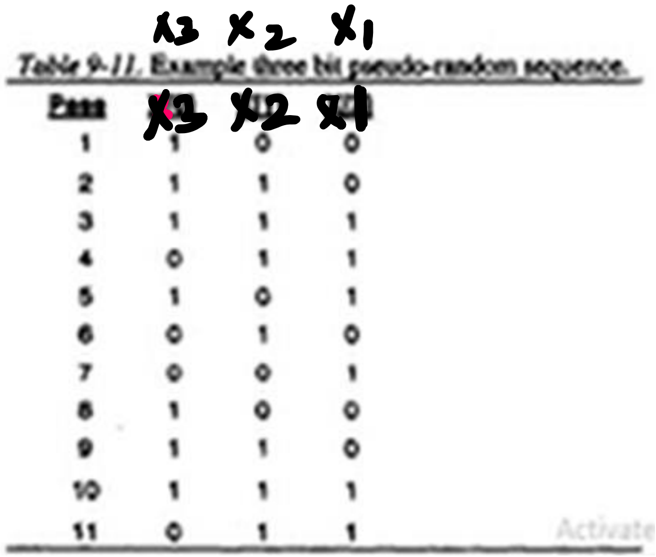
Linear Feedback Shift Register : It generates patterns of 0s and 1s which is random (preserves the randomness) and is deterministic

What is a primitive polynomial

- ✓ A polynomial eqn is the one which **has 1 and x^n with coefficient as 1.**
- ✓ These are **basically irreducible polynomial** (They are not divisible by any other polynomial other than 1 and itself)

Polynomial eqn : x^3+x+1 (n=3)





X3 X2 X1

Table 9-11. Example three bit pseudo-random sequence.

Pass	X3	X2	X1
1	1	0	0
2	1	1	0
3	1	1	1
4	0	1	1
5	1	0	1
6	0	1	0
7	0	0	1
8	1	0	0
9	1	1	0
10	1	1	1
11	0	1	1

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

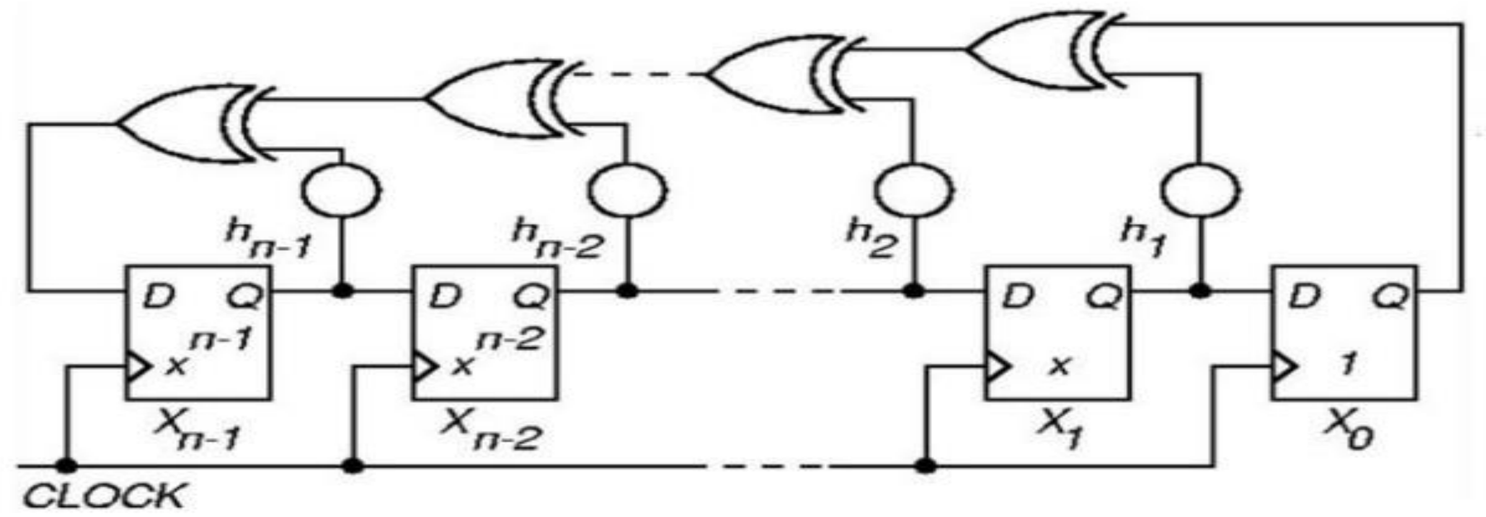
Built In Self Test (BIST)

Pattern Generation using BIST Engine

Pseudo Random BIST – Using LFSR

- LFSR employs series of n D FFs and XOR gates.
- The connection of DFF and XOR gates is based on polynomial eqn.

Polynomial is given as $f(x) = 1 + h_1x + h_2x^2 + \ldots\ldots\ldots h_{n-1}x^{n-1} + x^n$



What is a primitive polynomial

- ✓ A polynomial eqn is the one which has 1 and x^n with coefficient as 1.
- ✓ These are **basically irreducible polynomial** (They are not divisible by any other polynomial other than 1 and itself)

Ex: $f(x) = x^3 + x + 1$ for n=3

Is it a primitive polynomial ?

Yes- As it has 1 and x^n and it is irreducible.

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Pattern Generation using BIST Engine

Pseudo Random BIST – Using LFSR

Polynomial is given as $f(x) = 1 + h_1x + h_2x^2 + \ldots\ldots\ldots h_{n-1}x^{n-1} + x^n$

What is a primitive polynomial

- ✓ A polynomial eqn is the one which has 1 and x^n with coefficient as 1.
- ✓ These are **basically irreducible polynomial** (They are not divisible by any other polynomial other than 1 and itself)

Why primitive polynomials are used in pseudo random pattern generators?

The connection of DFF and XOR gates is based on polynomial eqn as

- The **primitive polynomial** ensures that all of **2^n -1 states are exercised,** where "n" is the number of latches that forms the LFSR

Example: The **primitive polynomial for a 3 bit LFSR** is **x^3+x+1**. That means that the taps for the XORs are on the X1 and X0 bits. The x3 and x terms corresponds to an XOR tap on X1 and X0 bits respectively

Note: The random pattern generated using primitive polynomial eqn does not included one state i.e., the **all zeros state** and special logic must be included in the LFSR if the all zeros state is required.

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

Pattern Generation using BIST Engine

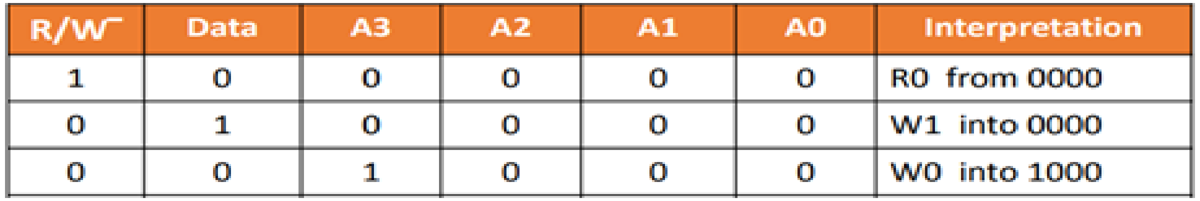
Pseudo Random BIST – Using LFSR

Pseudo-random patterns may be applied to some or all of a memory’s inputs.

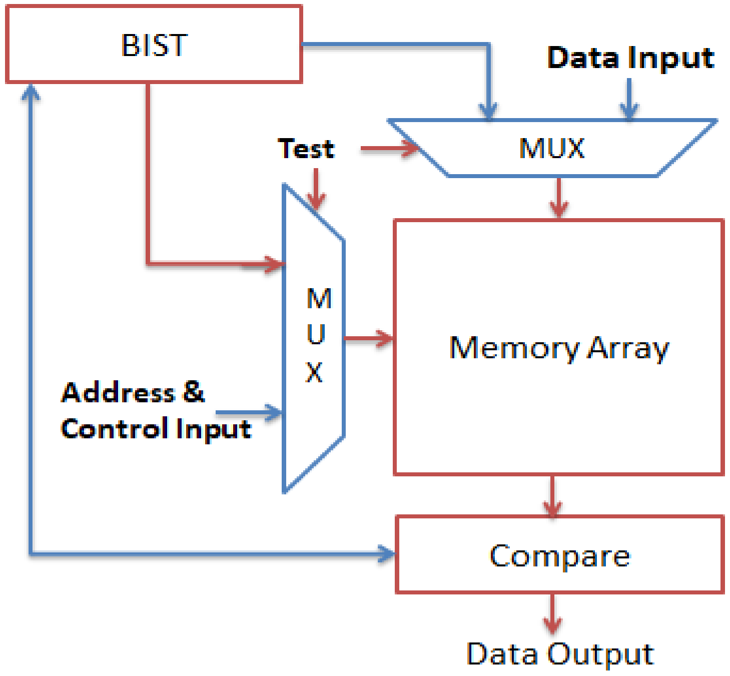
a) **Control Inputs:** If pseudo-random patterns are applied to the control inputs of a memory, the read and write operations happen pseudo-randomly.

b) **Address Input:** If these type patterns are applied to the address inputs of a memory, the addressing order is performed pseudo-randomly. In this case the addresses are proceeded through in one order; it could be referred to as “incrementing.”

c) **Data Input:** Pseudo-random stimuli can also be applied to the memory data inputs.



X5	X4	X3	X2	X1	X0	
R/$\overline{W}$	Data	A3	A2	A1	A0	Interpretation
1	0	0	0	0	0	R0 from 0000
0	1	0	0	0	0	W1 into 0000
0	0	1	0	0	0	W0 into 1000



Reference :“High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

Pattern Generation using BIST Engine

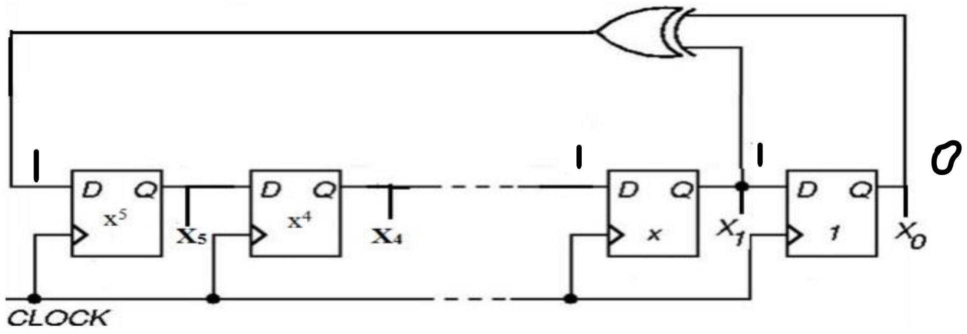
Pseudo Random BIST – Using LFSR

Polynomial is given as $f(x) = 1 + h_1x + h_2x^2 + \ldots\ldots\ldots h_{n-1}x^{n-1} + x^n$

Ex: **$f(x) = 1 + x^1 + x^6$**

h1 = 1, h2 = h3 = h4 = h5 = 0

6 FFs and 1 XOR gate with taps from 1 and x



The outputs of the LFSR, for the given polynomial, for successive clock cycles is shown

X 5	X4	X3	X2	X1	X0
1	0	0	0	0	0
0	1	0	0	0	0
0	0	1	0	0	0
0	0	0	1	0	0
0	0	0	0	1	0
1	0	0	0	0	1
1	1	0	0	0	0
0	1	1	0	0	0
0	0	1	1	0	0
0	0	0	1	1	0
1	0	0	0	1	1

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

Pattern Generation using BIST Engine

Pseudo Random BIST – Using LFSR

Connecting the outputs to different address and control signals, the pseudo random patterns can be interpreted as follows

$R/\overline{W}$	Data	A3	A2	A1	A0	Interpretation
1	0	0	0	0	0	R0 from 0000
0	1	0	0	0	0	W1 into 0000
0	0	1	0	0	0	W0 into 1000
0	0	0	1	0	0	W0 into 0100
0	0	0	0	1	0	W0 into 0010
1	0	0	0	0	1	R0 from 0001
1	1	0	0	0	0	R1 from 0000
0	1	1	0	0	0	W1 into 1000
0	0	1	1	0	0	W0 into 1100
0	0	0	1	1	0	W0 into 0110
1	0	0	0	1	1	R0 from 0011

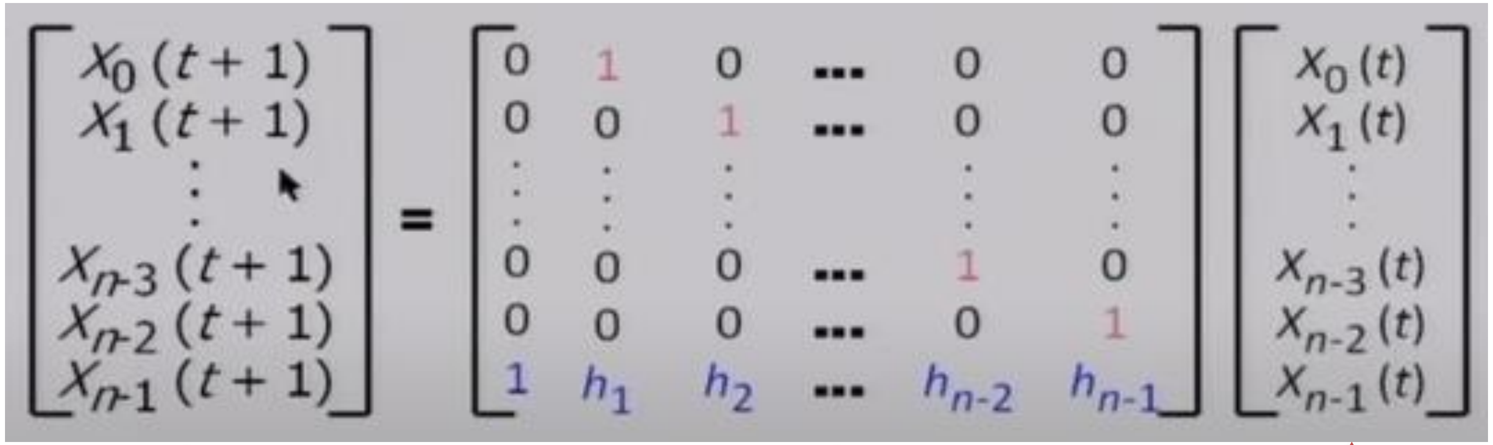
Reference :“High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

Pattern Generation using BIST Engine

Pseudo Random BIST – Using LFSR

The Standard LFSR can be expressed in the form of Matrix as follows



$$\begin{bmatrix} X_0(t+1)\\ X_1(t+1)\\ \vdots\\ X_{n-3}(t+1)\\ X_{n-2}(t+1)\\ X_{n-1}(t+1)\end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0\\ 0 & 0 & 1 & \cdots & 0 & 0\\ \vdots & \vdots & \vdots & & \vdots & \vdots\\ 0 & 0 & 0 & \cdots & 1 & 0\\ 0 & 0 & 0 & \cdots & 0 & 1\\ 1 & h_1 & h_2 & \cdots & h_{n-2} & h_{n-1}\end{bmatrix} \begin{bmatrix} X_0(t)\\ X_1(t)\\ \vdots\\ X_{n-3}(t)\\ X_{n-2}(t)\\ X_{n-1}(t)\end{bmatrix}$$

New output

Companion Matrix (Ts)

Current Values in FF

X(t+1) = Ts. X(t)

Properties of Ts

1. In the First Column Last bit is 1, this comes from feedback of the first feedback to the Last FF.
2. The h1 to hn-1 in the nth Row can be either 1s or 0s, this depends on whether the values are tapped or NOT.
3. The Rest of the Matrix is identity Matrix.

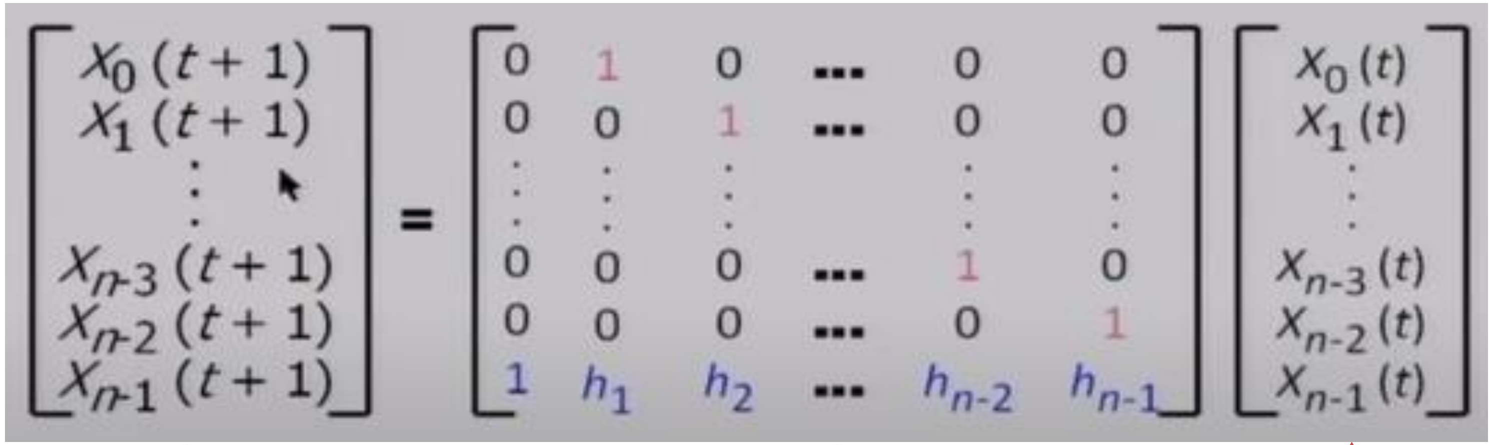
Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

Pattern Generation using BIST Engine

Pseudo Random BIST – Using LFSR

The Standard LFSR can be expressed in the form of Matrix as follows



$$\begin{bmatrix} X_0(t+1) \\ X_1(t+1) \\ \vdots \\ X_{n-3}(t+1) \\ X_{n-2}(t+1) \\ X_{n-1}(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ 1 & h_1 & h_2 & \cdots & h_{n-2} & h_{n-1} \end{bmatrix} \begin{bmatrix} X_0(t) \\ X_1(t) \\ \vdots \\ X_{n-3}(t) \\ X_{n-2}(t) \\ X_{n-1}(t) \end{bmatrix}$$

New output

Companion Matrix (Ts)

Current Values in FF

X(t+1) = Ts. X(t)

Properties of Ts

1. In the First Column Last bit is 1, this comes from feedback of the first feedback to the Last FF.
2. The h1 to hn-1 in the nth Row can be either 1s or 0s, this depends on whether the values are tapped or NOT.
3. The Rest of the Matrix is identity Matrix.

Ex: **$f(x) = 1 + x^1 + x^6$**

h1 = 1, h2 = h3 = h4 = h5 = 0

Reference :“High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

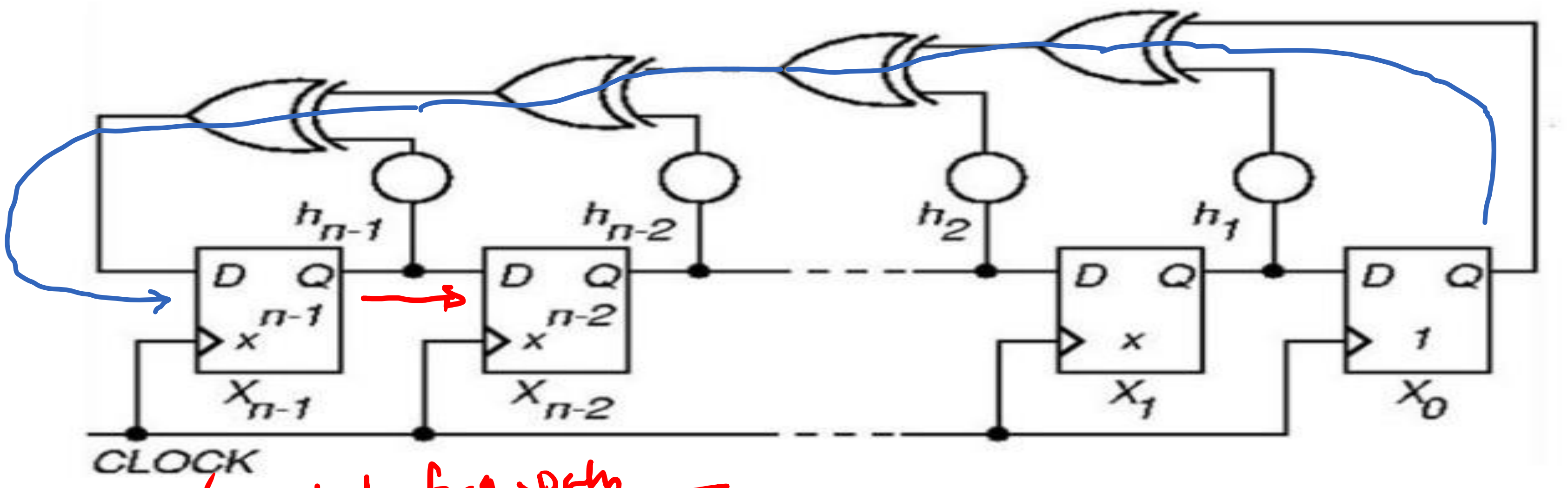
Built In Self Test (BIST)

Pattern Generation using BIST Engine

Pseudo Random BIST – Using LFSR

Drawback of LFSR as Pseudo Random Pattern Generator

The Standard LFSR may have worst speed if tapping is done all the points as ‘ n’ XOR gates will come in feedback between Xn-1 and X0. i.e., X0 has to propagate through n XOR gates to reach input of Xn-1



slower

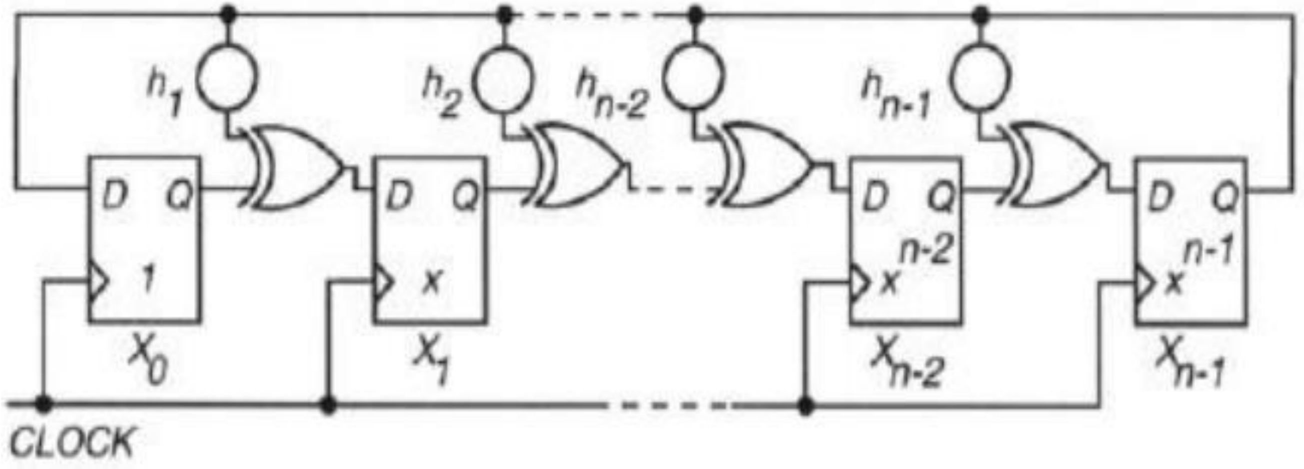
✓ clock freq path

Reference :“High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Pattern Generation using BIST Engine

Pseudo Random BIST – Using Modular LFSR

Polynomial is given as $f(x) = 1 + h1.x + h2.x^2 + \ldots\ldots\ldots hn\text{-}1.x^{n-1} + x^n$



Observations:

1. XOR gates are placed in the feedback for ALL the paths i.e, between previous output and Next input.
2. These XOR gate are Internal and the coefficient value indicates the presence of XOR gate.If Coefficient is ‘1’, XOR is present else output of previous DFF is directly connected as Input to next DFF.
3. Feedback from Left to Right
4. It also produces the same sequence.
5. In this Companion Matrix $Tm=Ts^T$
6. In this

   X(t+1) = Tm. X(t)

Reference :“High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

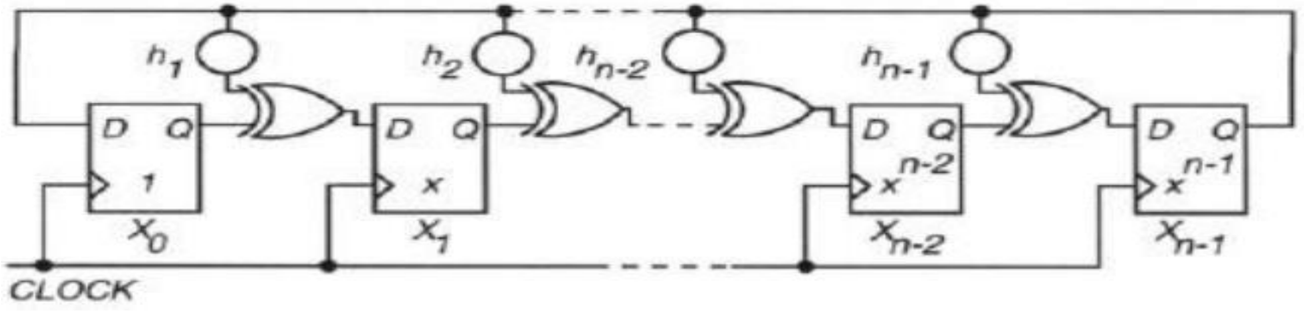
Built In Self Test (BIST)

Pattern Generation using BIST Engine

Pseudo Random BIST – Using Modular LFSR

Polynomial is given as $f(x) = 1 + h1.x + h2.x^2 + \ldots\ldots\ldots hn\text{-}1.x^{n-1} + x^n$

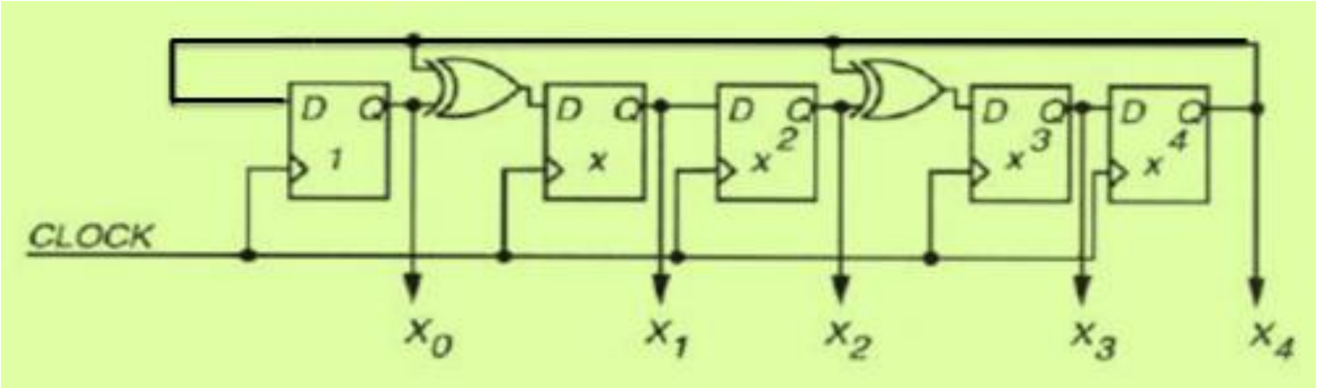
Ex: $f(x) = 1+x+x^3+x^5$

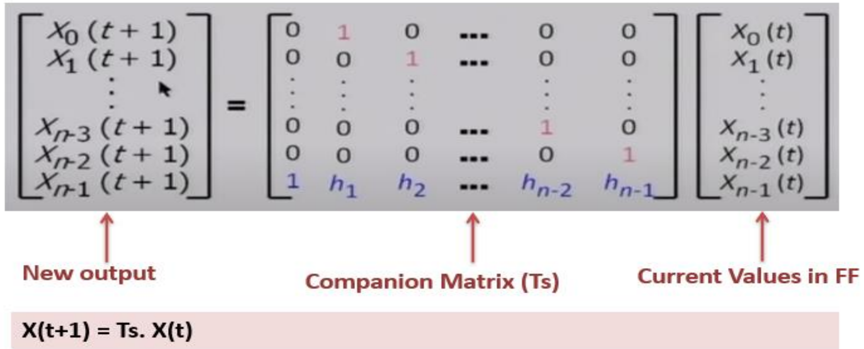


Observations:

1. XOR gates are placed in the feedback for ALL the paths i.e, between previous output and Next input.

X(t+1) = Tm. X(t)





$$\begin{bmatrix} X_0(t+1) \\ X_1(t+1) \\ \vdots \\ X_{n-3}(t+1) \\ X_{n-2}(t+1) \\ X_{n-1}(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ 1 & h_1 & h_2 & \cdots & h_{n-2} & h_{n-1} \end{bmatrix} \begin{bmatrix} X_0(t) \\ X_1(t) \\ \vdots \\ X_{n-3}(t) \\ X_{n-2}(t) \\ X_{n-1}(t) \end{bmatrix}$$

New output

Companion Matrix (Ts)

Current Values in FF

X(t+1) = Ts. X(t)

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

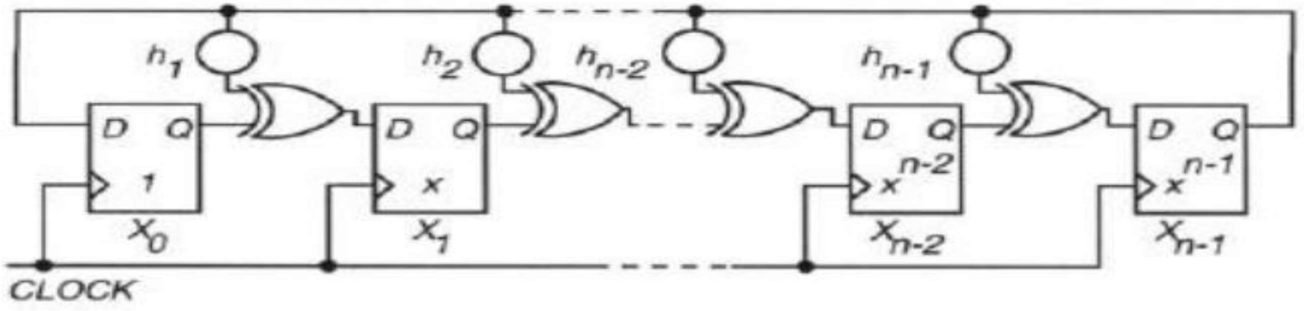
Built In Self Test (BIST)

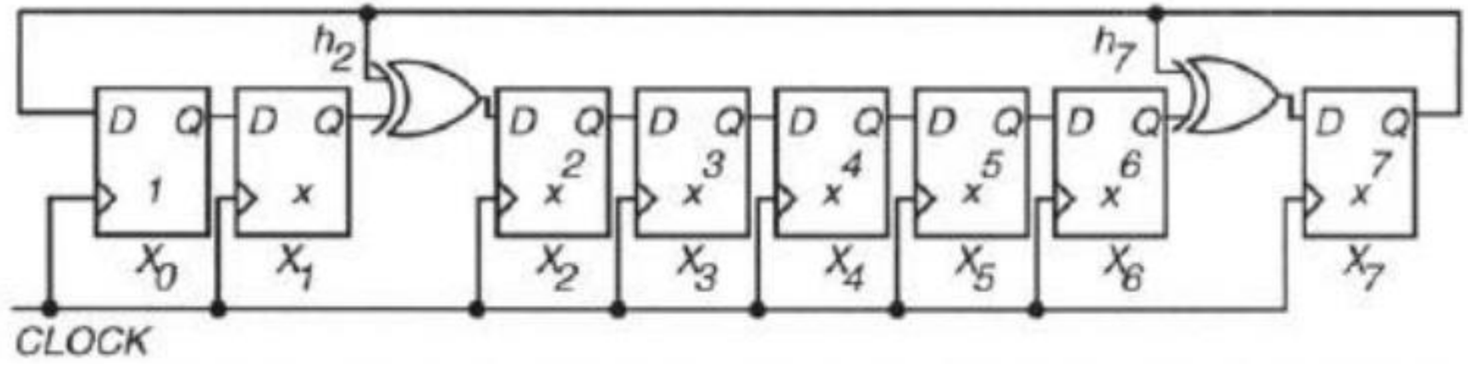
Pattern Generation using BIST Engine

Pseudo Random BIST – Using Modular LFSR

Polynomial is given as $f(x) = 1 + h1.x + h2.x^{2} + ... hn\text{-}1.x^{n-1} + x^{n}$

Ex: $f(x) = 1 + x^{2} + x^{7} + x^{8}$





Observations:

1. XOR gates are placed in the feedback for ALL the paths i.e, between previous output and Next input.

X(t+1) = Tm. X(t)

Advantage:

Modular LFSR is Faster compared to conventional LFSR as the XOR gates are NOT between X0 and Xn-1

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

Pattern Generation using BIST Engine

Comparison

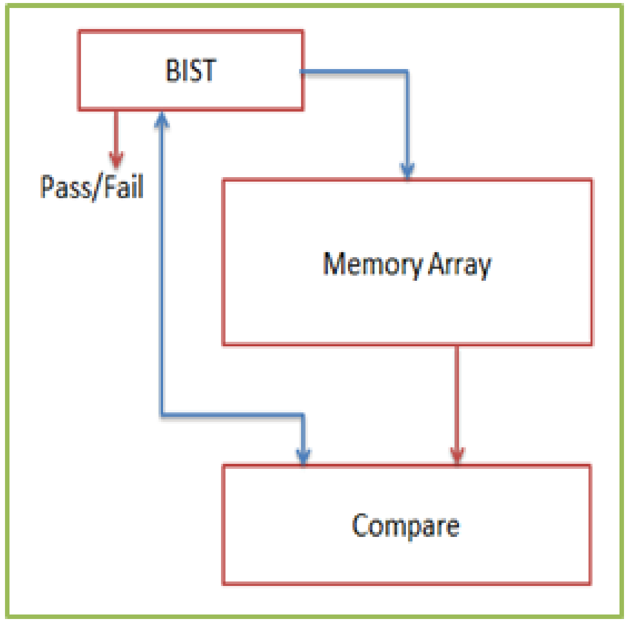
How to collect the and compare with the standard response ?

The standard response can be obtained from simulation as we know pseudo random test patterns as they are repeatable.

Where to store these standard patterns (golden patterns) obtained for fault free Memory?

- ✓ **Can be stored in ROM and**
- ✓ **Requires Large size ROM to store the standard response which means Larger Area required for storing the standard response pattern**

To reduce the requirement of ROM, It is necessary to reduce the Volume of data to be stored



Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

Pattern Generation using BIST Engine

i) Std LFSR

ii) Modular LFSR

pseudo rand pattern

ROM

BIST Response Compaction

Compaction: ✓

- ✓ Method of drastically reducing the number of bits in the original circuit response during testing in which some information is lost.
- ✓ It is Non reversible

Signature:

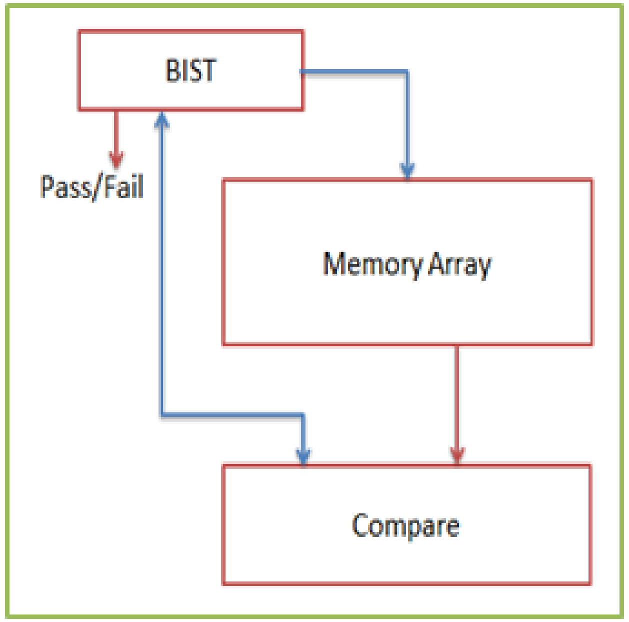
- ✓ The number computed after compaction of response

Signature Analysis :

- ✓ Comparison of the good memory signature and the obtained memory signature to determine if the memory is faulty

Aliasing :

- ✓ During circuit response compaction, because of information loss, it is possible that **signature of bad machine may match the good machine signature.**
- ✓ In such cases, a failing circuit will pass the testing process

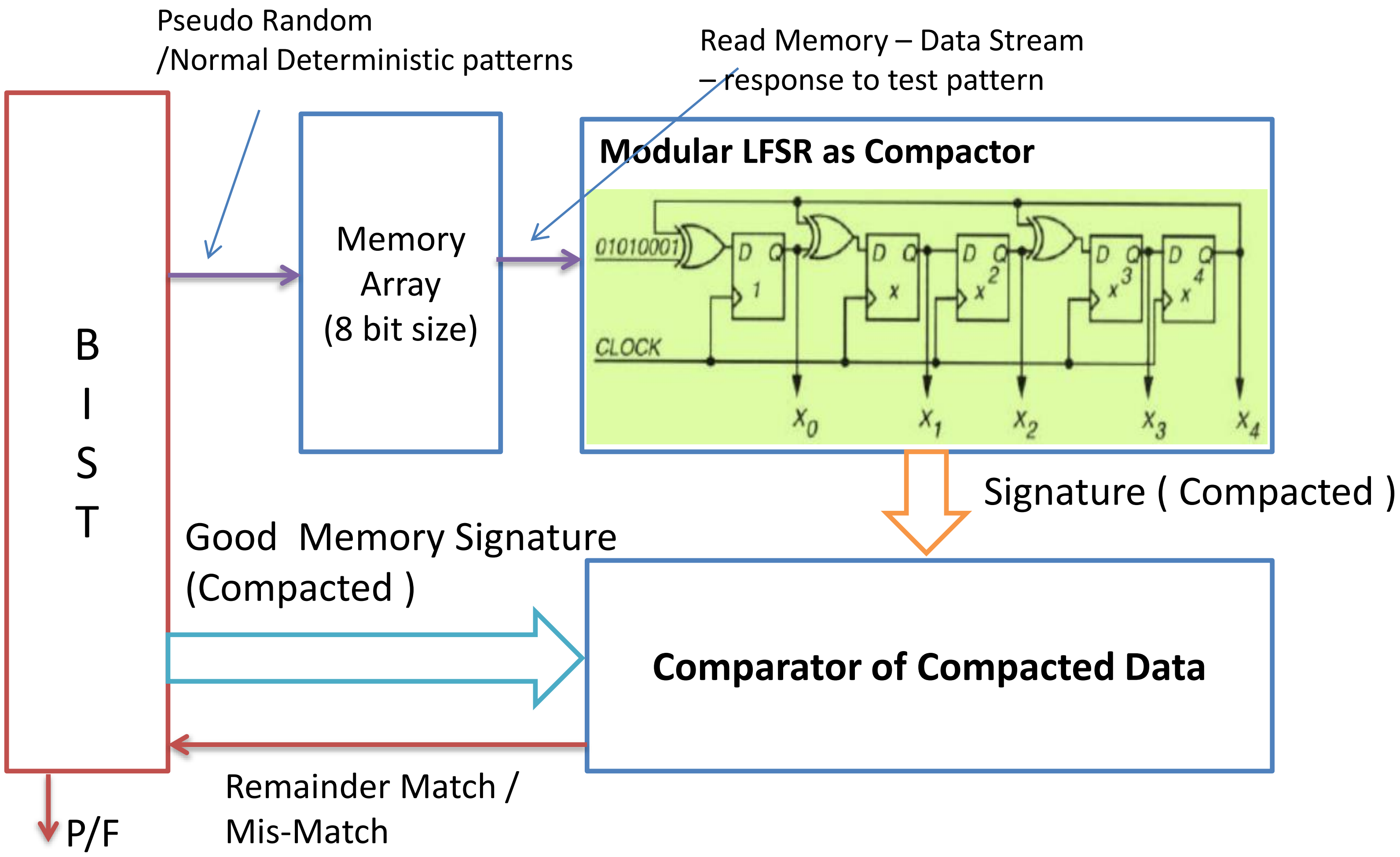


Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

Pattern Generation using BIST Engine

Modular LFSR as Response Compaction



Pseudo Random /Normal Deterministic patterns

Read Memory – Data Stream – response to test pattern

B I S T

Memory Array (8 bit size)

Modular LFSR as Compactor

Signature (Compacted)

Good Memory Signature (Compacted)

Comparator of Compacted Data

Remainder Match / Mis-Match

P/F

Reference :“High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

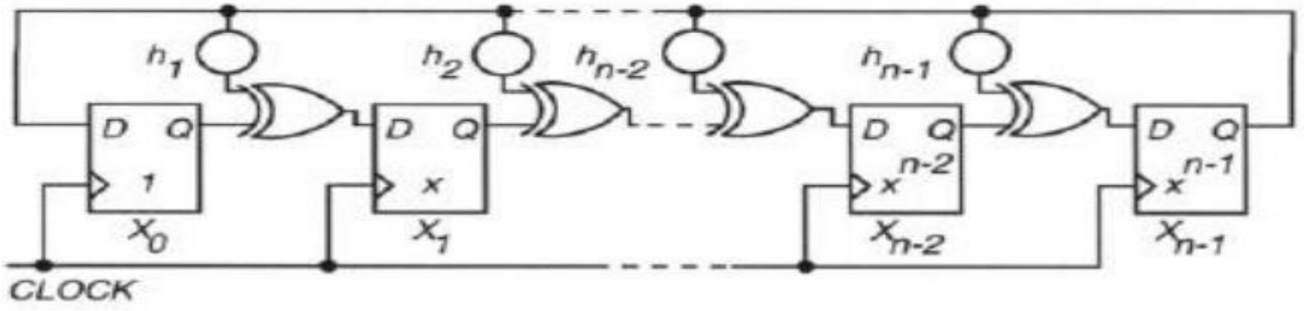
Pattern Generation using BIST Engine

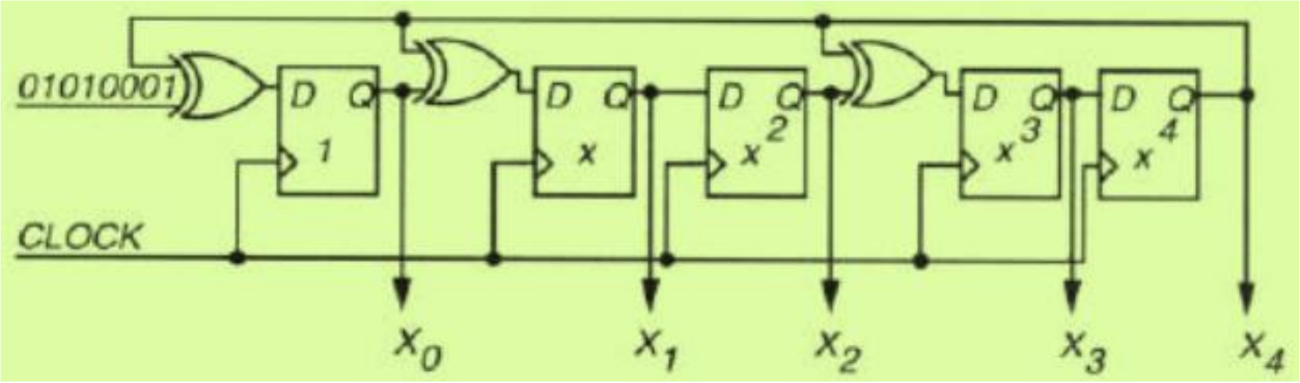
Modular LFSR as Response Compaction

Polynomial is given as **f(x) = 1 + h1.x +h2.x^2 +………hn-1.x^{n-1}+x^n**

Ex: f(x) = $1+x+x^3+x^5$

Use the eqn x(t+1) = Tm . X(t)





Response Compaction:

- The characteristic equation is f(x) = $1+x+x^3+x^5$
- At the **X0 input a additional XOR gate** is added to Modular LFSR circuit designed for given characteristic equation.
- Through this **XOR gate the an 8-bit data read from memory is given as Input** i.e., Input bit stream Ex: 01010001.
- The outputs of the Response compacter after all bits are fed is obtained.

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

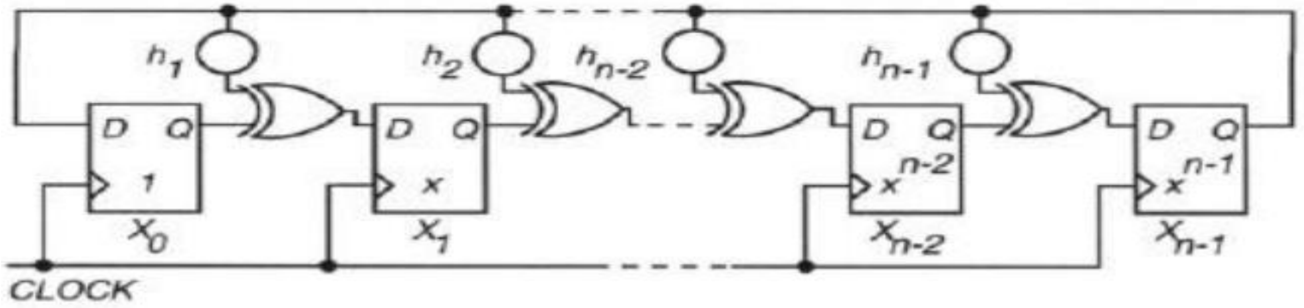
Built In Self Test (BIST)

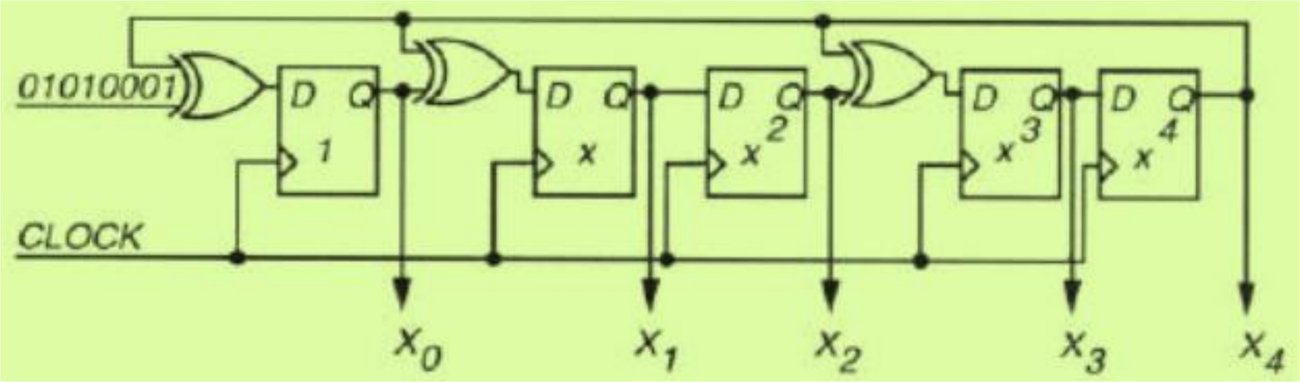
Pattern Generation using BIST Engine

Modular LFSR as Response Compaction

Polynomial is given as $f(x) = 1 + h1.x + h2.x^2 + \ldots\ldots\ldots hn\text{-}1.x^{n-1} + x^n$

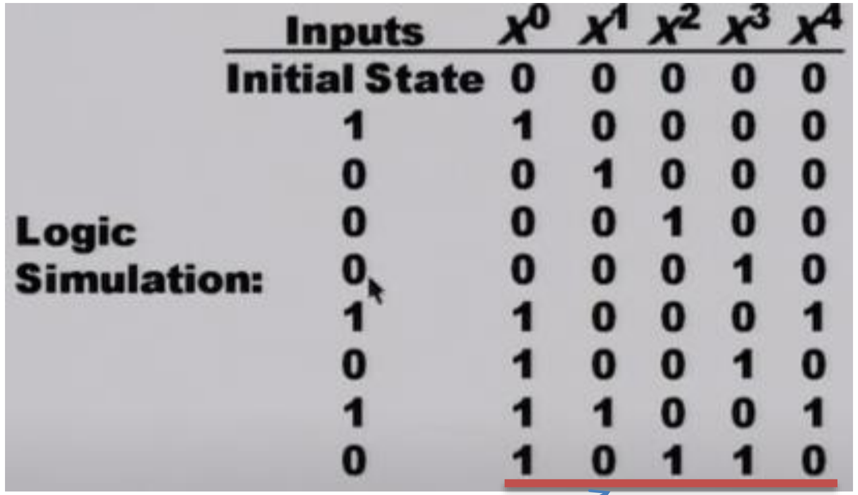
Ex: $f(x) = 1+x+x^3+x^5$





Response Compaction:

- The outputs of the Response compacter is



	Inputs	x^0	x^1	x^2	x^3	x^4
	Initial State	0	0	0	0	0
Logic Simulation:	1	1	0	0	0	0
	0	0	1	0	0	0
	0	0	0	1	0	0
	0	0	0	0	1	0
	1	1	0	0	0	1
	0	1	0	0	1	0
	1	1	1	0	0	1
	0	1	0	1	1	0

Remainder

Remainder Polynomial: $1+X^2+X^3$

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

Pattern Generation using BIST Engine

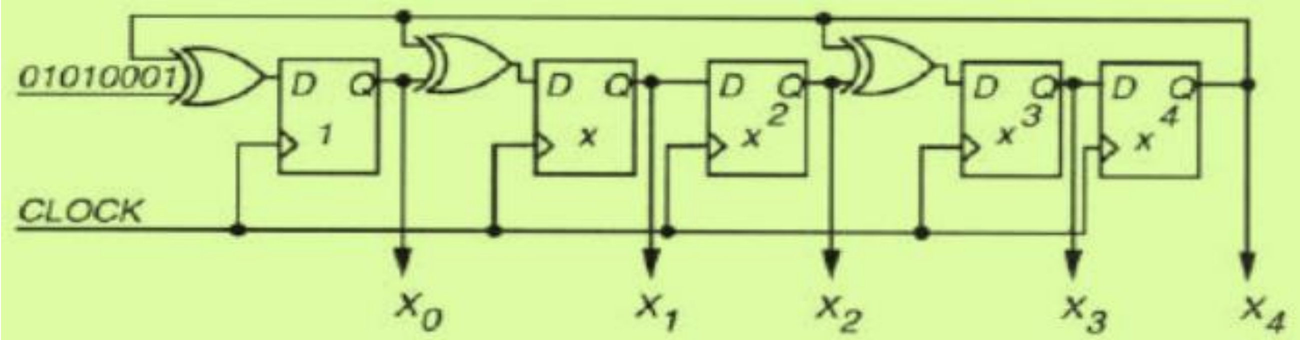
0 1 0 1 0 0 0 1

x^0 x^1 x^2 x^3 x^4 x^5 x^6 x^7

$\underbrace{x^1+x^3+x^7}$

$\hookrightarrow x^7+x^3+x^1$

Modular LFSR as Response Compaction



Characteristic Polynomial :f(x) = $x^5 + x^3 + x + 1$

What Modular LFSR as Response Compactor does ?

In this method,

- ✓ the circuit output data stream is **treated as a descending order coefficient polynomial.**

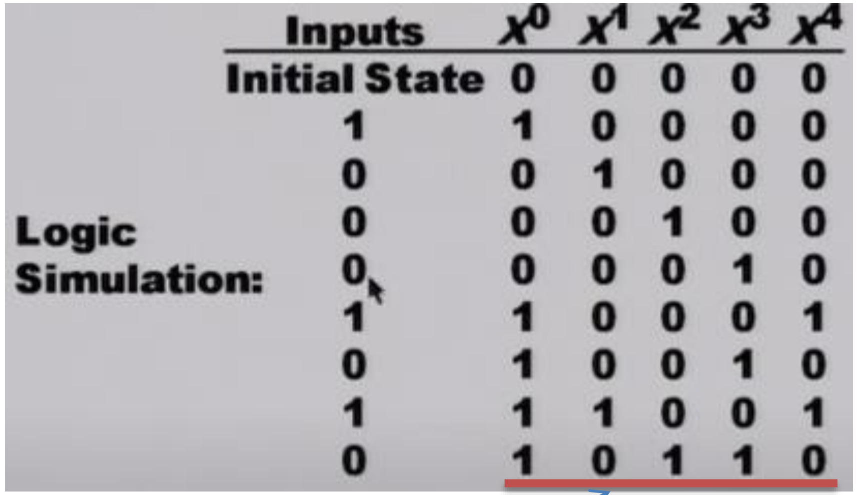
  Ex: 01010001 = $0.x^0+1.x^1+0x^2+1.x^3+0.x^4+0.x^5+0.x^6+1x^7$

  = $x+x^3+x^7$

- ✓ The response compacter LFSR performs polynomial division of this data stream polynomial by the characteristic polynomial of the LFSR.
- ✓ The final state of the Modular LFSR is the polynomial remainder of this division.

Response Compaction:

- The outputs of the Response compacter is



	Inputs	x^0	x^1	x^2	x^3	x^4
	Initial State	0	0	0	0	0
Logic Simulation:	1	1	0	0	0	0
	0	0	1	0	0	0
	0	0	0	1	0	0
	0	0	0	0	1	0
	1	1	0	0	0	1
	0	1	0	0	1	0
	1	1	1	0	0	1
	0	1	0	1	1	0

Remainder

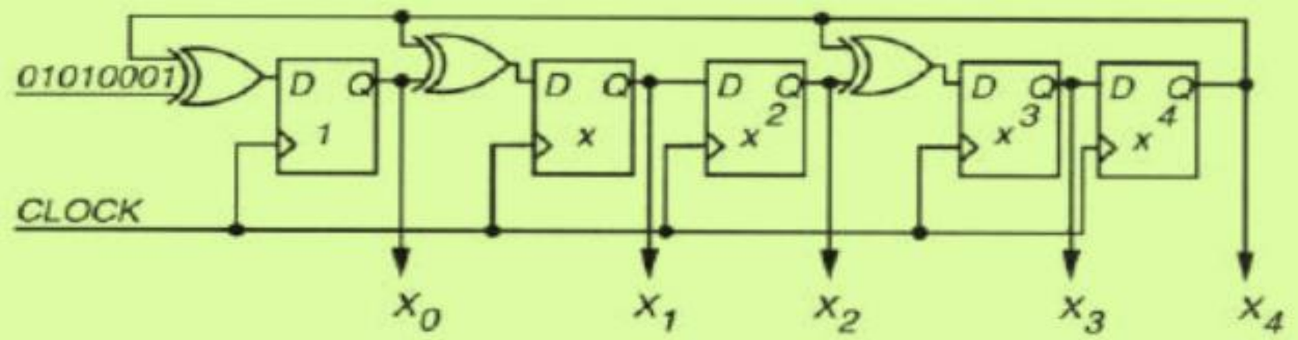
Remainder Polynomial: $1+X^2+X^3$

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

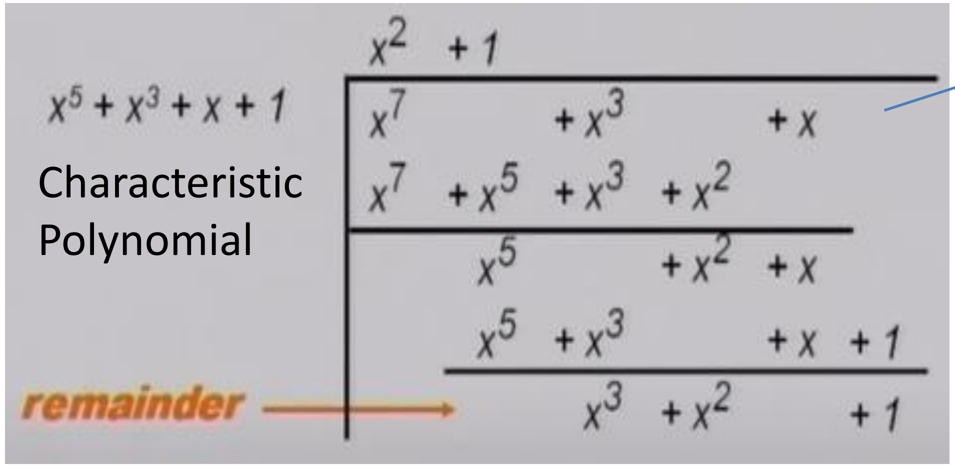
Pattern Generation using BIST Engine

Modular LFSR as Response Compaction



What Modular LFSR as Response Compactor does ?

In this method,



$x^2 + 1$

$x^5 + x^3 + x + 1$ | $x^7 + x^3 + x$

$x^7 + x^5 + x^3 + x^2$

$x^5 + x^2 + x$

$x^5 + x^3 + x + 1$

remainder → $x^3 + x^2 + 1$

Characteristic Polynomial

Input Data polynomial

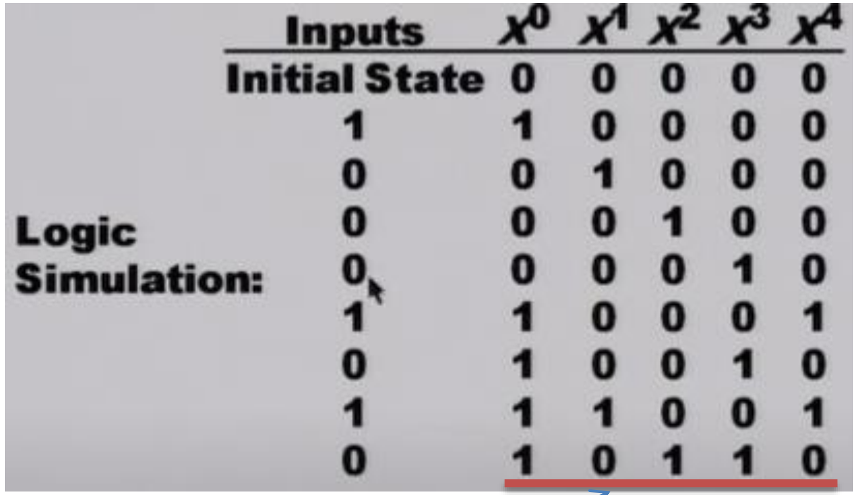
X7 XOR X7 =1 XOR 1=0

X3 XOR X3 = 1 XOR 1 =0

Characteristic Polynomial :f(x) = $x^5 + x^3 + x + 1$

Response Compaction:

- The outputs of the Response compacter is



	Inputs	x^0	x^1	x^2	x^3	x^4
	Initial State	0	0	0	0	0
Logic Simulation:	1	1	0	0	0	0
	0	0	1	0	0	0
	0	0	0	1	0	0
	0	0	0	0	1	0
	1	1	0	0	0	1
	0	1	0	0	1	0
	1	1	1	0	0	1
	0	1	0	1	1	0

Remainder

Remainder Polynomial: **$1+X^2+X^3$**

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Pattern Generation using BIST Engine

Modular LFSR as Response Compaction

Characteristic Polynomial : $f(x) = 1 + x + x^3$

Data stream:

a) 01010

b) 10110

c) 00010

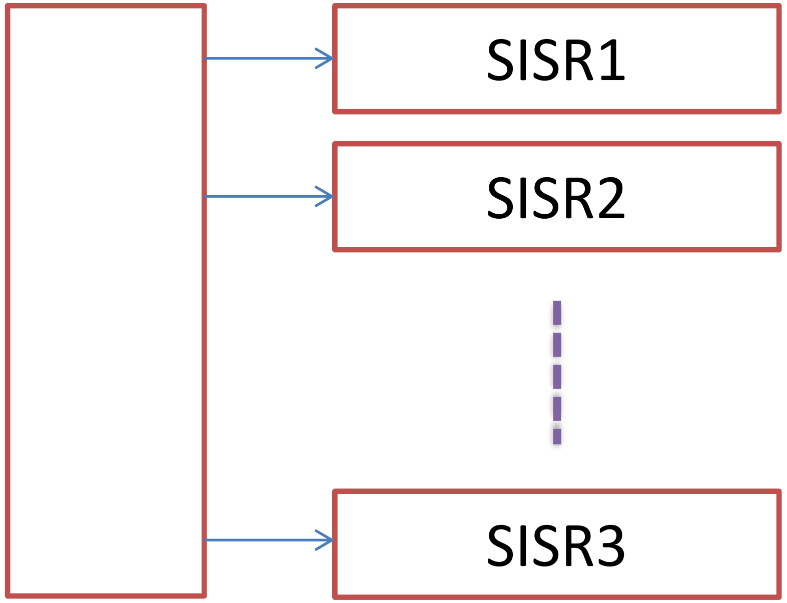
Built In Self Test (BIST)

Pattern Generation using BIST Engine

Modular LFSR as Response Compaction

Drawbacks:

1. For every primary Output need to have ONE LFSR or One Signature Register which is known as **Single Input Signature Register (SISR).**
2. It means for entire test sequence – Multiple SISR are required. This increases the hardware overhead.



What if the normal deterministic entire test patterns are applied to memory inputs ?

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

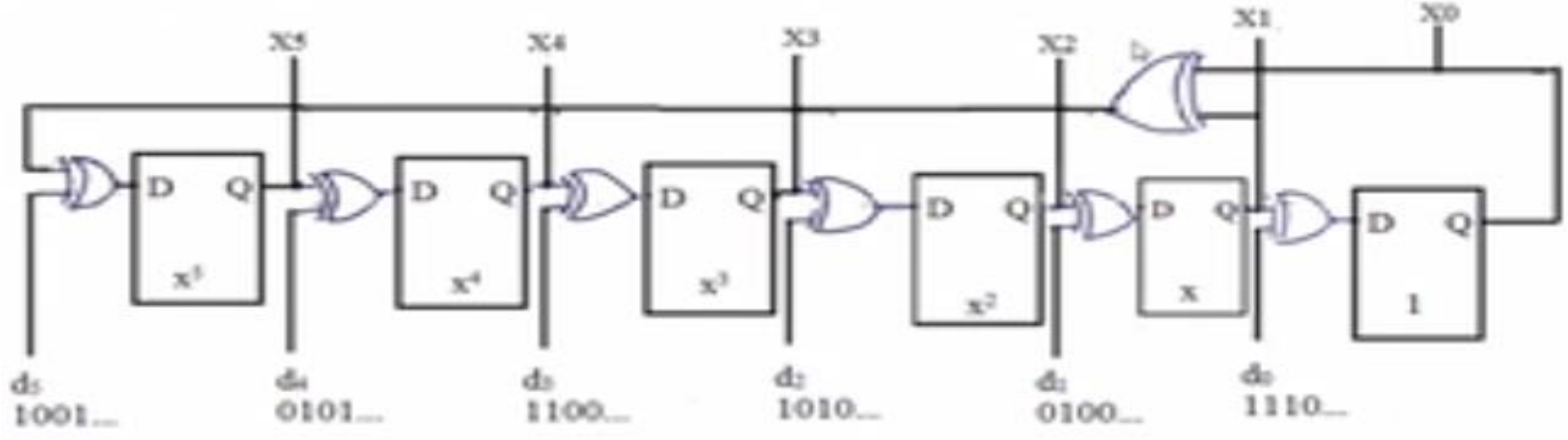
Built In Self Test (BIST)

Multiple Input Signature Register - MISR

- MISR is used to observe memory outputs when **normal deterministic test patterns are applied to memory inputs**
- It captures the results of all the test cycles and accumulates a signature
- Cannot provide cycle-by-cycle pass/fail result
- If the signature at the end of the test matches the simulated good memory result, the memory passes the test
- **Drawback:** Aliasing can occur Errors in 2 cycles can be nullified and may result in a good signature since MISR does not provide the result after each cycle.

signature
set of pattern
↳ single sign

MISR using standard LFSR



Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

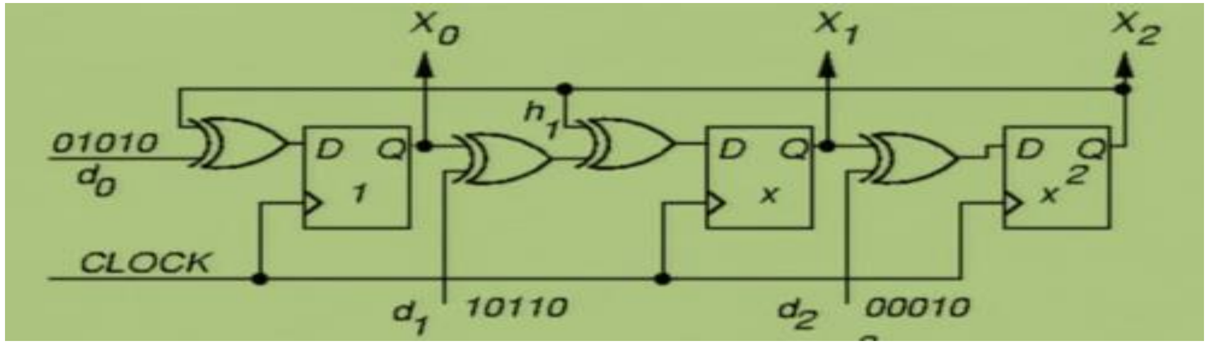
Built In Self Test (BIST)

Multiple Input Signature Register - MISR

Circuit of MISR for the characteristic polynomial f(x) = 1 + x + x^3

Entire Data Stream:

- ✓ 01010
- ✓ 10110
- ✓ 00010



- An MISR takes the data output values read from the **memory during the entire test sequence and generates a string of bits after Compaction**
- MISR is identical to a **Modular LFSR except that an additional XOR gate is present at the input of each FF**

$$\begin{bmatrix} X_0(t+1) \\ X_1(t+1) \\ X_2(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_0(t) \\ X_1(t) \\ X_2(t) \end{bmatrix} + \begin{bmatrix} d_0(t) \\ d_1(t) \\ d_2(t) \end{bmatrix}$$

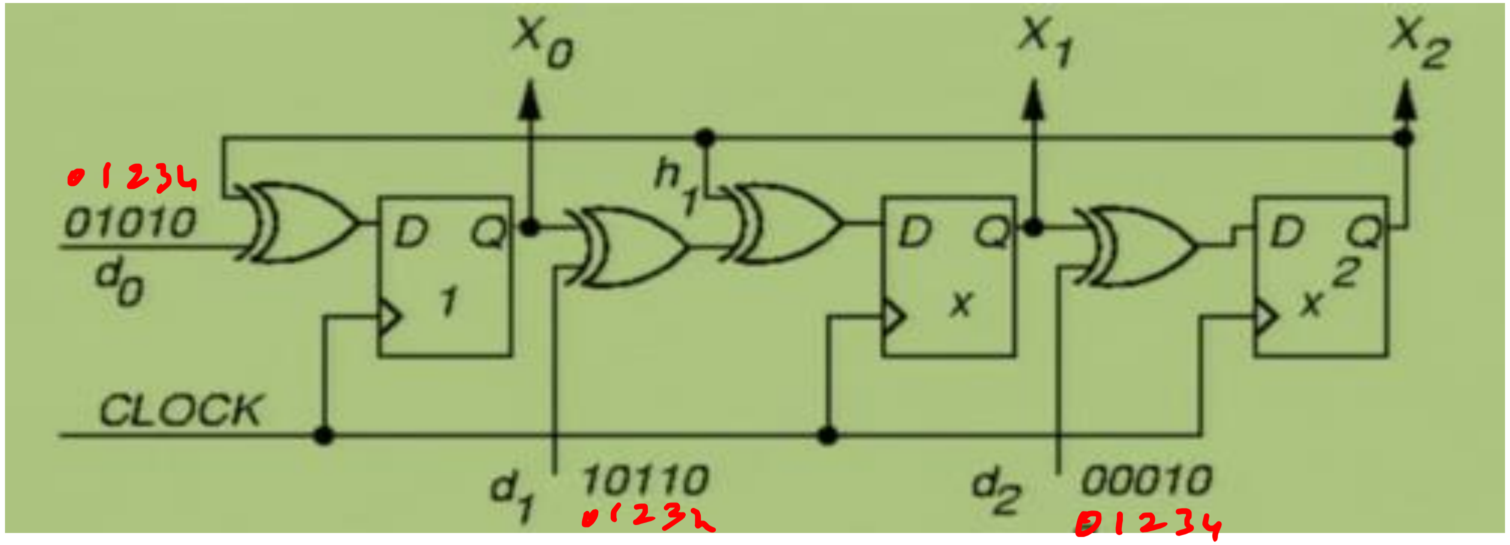
Reference :“High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

Multiple Input Signature Register - MISR

- An MISR takes the data output values read from the **memory during the entire test sequence and generates a string of bits after Compaction**
- MISR is identical to a **Modular LFSR except that an additional XOR gate is present at the input of each FF**

Circuit of MISR for the characteristic polynomial f(x) = 1 + x + x^3



✓ The additional XOR gates helps in giving Multiple data streams from different locations as Input to MISR

✓ In the example, As there are 3 FF, Simultaneously 3 data streams are given as inputs

MISR is used to observe memory outputs when **normal deterministic test patterns are applied to memory inputs**

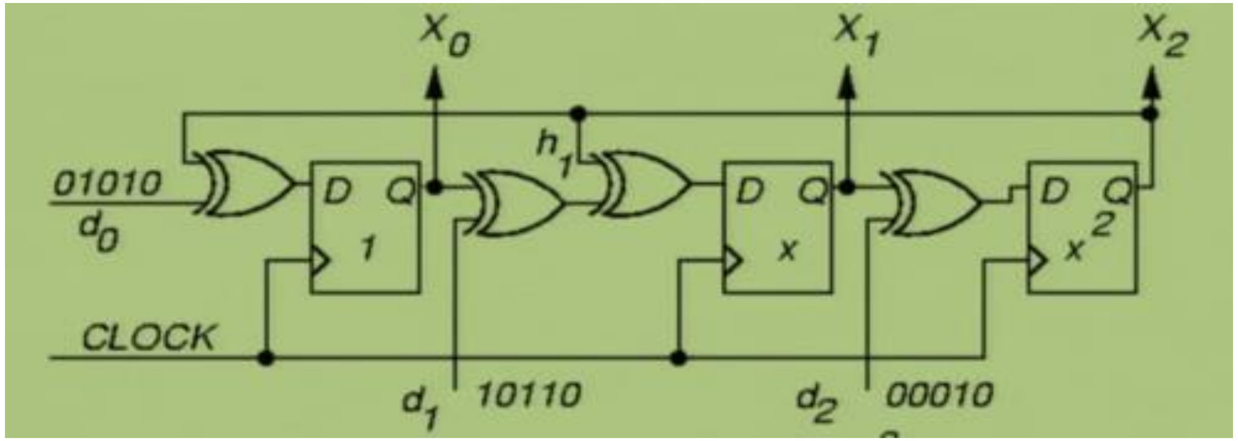
Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

Multiple Input Signature Register - MISR

The resulting **signature is the XOR of the 3 individual signatures obtained by the polynomial division** of each of the inputs by the characteristic polynomial

Circuit of MISR for the characteristic polynomial f(x) = 1 + x + x^3



The circuit implements the following equation system

$$\begin{bmatrix} X_0(t+1) \\ X_1(t+1) \\ X_2(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_0(t) \\ X_1(t) \\ X_2(t) \end{bmatrix} + \begin{bmatrix} d_0(t) \\ d_1(t) \\ d_2(t) \end{bmatrix}$$

X0 (t+1) = X2 (t) + d0(t)
X1 (t+1) = X0 (t) + X2 (t) + d1(t)
X2 (t+1) = X1 (t) + d2 (t)

	$x_0(t+1)$ $= x_2(t) + d_0(t)$	$x_1(t+1) =$ $x_0(t) + x_2(t) + d_1(t)$	$x_2(t+1)$ $x_1(t) + d_2(t)$
	$x_2(t)$ $d_0(t)$ o/p $0 \oplus 0 = 0$ ✓	$x_0(t)$ $x_2(t)$ $d_1(t)$ o/p $0 \oplus 0 + 0 = \textcircled{0}$	$x_1(t)$ $d_2(t)$ o/p $0 \oplus 0 = \textcircled{0}$
	$0 \oplus 1 = 1$	$0 \oplus 0 \oplus 1 = 1$	$0 \oplus 1 = 1$
	$1 \oplus 0 = 1$	$1 \oplus 1 \oplus 1 = 1$	$1 \oplus 0 = 1$
	$1 \oplus 1 = 0$	$1 \oplus 1 \oplus 0 = 0$	$1 \oplus 0 = 1$
	$1 \oplus 0 = 1$	$0 \oplus 1 \oplus 1 = 0$	$0 \oplus 0 = 0$
Remainder	✓ $\boxed{1}$	$\boxed{0}$	$\boxed{0}$
	Remainder Poly = 1		

+ represents XOR operation

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

Multiple Input Signature Register - MISR

Circuit of MISR for the characteristic polynomial f(x) = 1 + x + x^3

X0 (t+1) = X2 (t) + d0(t)

X1 (t+1) = X0 (t) + X2 (t) + d1(t)

X2 (t+1) = X1 (t) + d2 (t)

$$\begin{bmatrix} X_0(t+1) \\ X_1(t+1) \\ X_2(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_0(t) \\ X_1(t) \\ X_2(t) \end{bmatrix} + \begin{bmatrix} d_0(t) \\ d_1(t) \\ d_2(t) \end{bmatrix}$$

- ✓ Assuming a 5-bit data, lets say the data read from the memory in each cycle is as follows:
  - 1 st cycle – 01010
  - 2nd cycle –10110
  - 3rd cycle – 00010
- ✓ The resulting **signature is the XOR of the 3 individual signatures obtained by the polynomial division** of each of the inputs by the characteristic polynomial

	$x_0(t+1) = x_2(t) + d_0(t)$	$x_1(t+1) = x_0(t) + x_2(t) + d_1(t)$	$x_2(t+1)$ $x_1(t) + d_2(t)$
	$x_2(t)$ $d_0(t)$ o/p 0 ⊕ 0 = 0 ✓	$x_0(t)$ $x_2(t)$ $d_1(t)$ o/p 0 ⊕ 0 + 0 = 0	$x_1(t)$ $d_2(t)$ o/p 0 ⊕ 0 = 0
	0 ⊕ 1 = 1	0 ⊕ 0 ⊕ 1 = 1	0 ⊕ 1 = 1
	1 ⊕ 0 = 1	1 ⊕ 1 ⊕ 1 = 1	1 ⊕ 0 = 1
	1 ⊕ 1 = 0	1 ⊕ 1 ⊕ 0 = 0	1 ⊕ 0 = 1
	1 ⊕ 0 = 1	0 ⊕ 1 ⊕ 1 = 0	0 ⊕ 0 = 0
Remainder	$\boxed{1}$	$\boxed{0}$	$\boxed{0}$
	Remainder	Poly = 1	

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

Multiple Input Signature Register - MISR

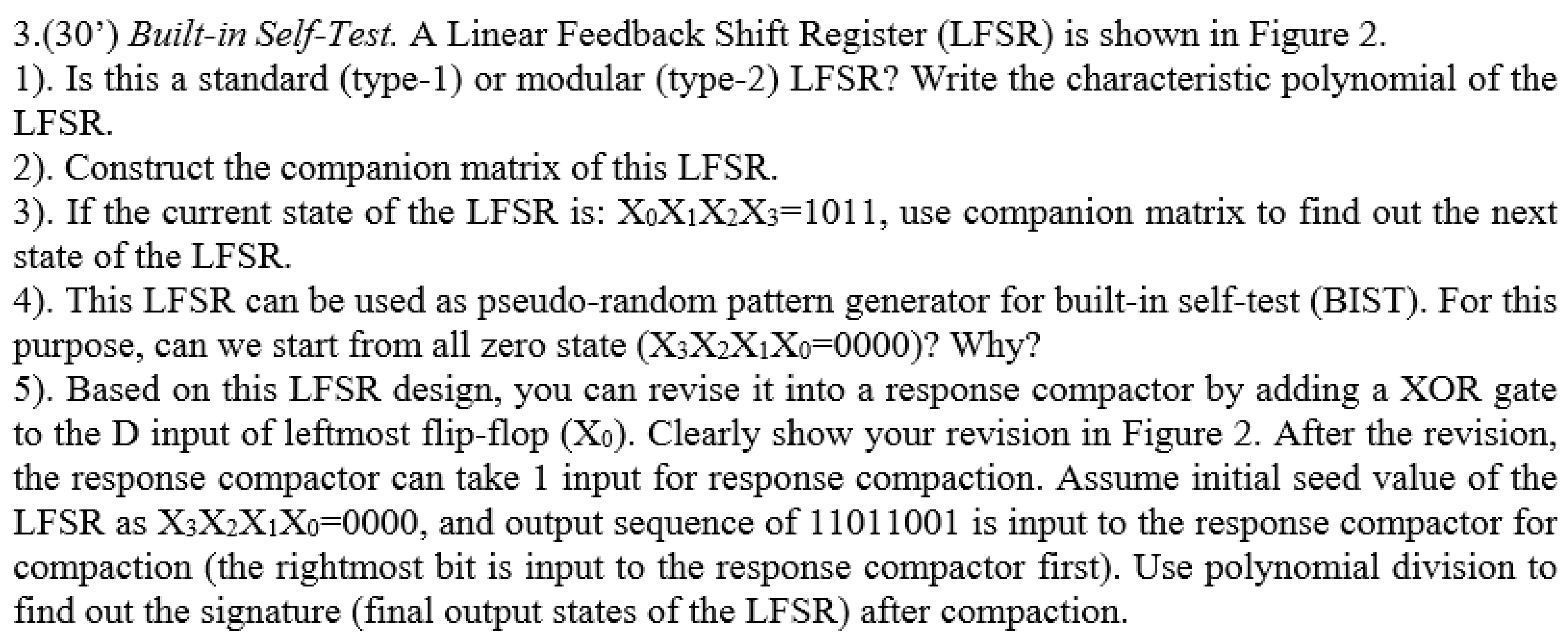
a) 01010 = $0 \cdot x^0 + 1x^1 + 0x^2 + 1x^3 + 0x^4 = x + x^3 \Rightarrow x^3 + x$

b) 10110 =

c) 00010 =

$$x^3+x+1 \overline{\smash{)}\, x^3} \quad 1$$

$$\underline{x^3}$$



3.(30') *Built-in Self-Test.* A Linear Feedback Shift Register (LFSR) is shown in Figure 2.

1). Is this a standard (type-1) or modular (type-2) LFSR? Write the characteristic polynomial of the LFSR.

2). Construct the companion matrix of this LFSR.

3). If the current state of the LFSR is: $X_0X_1X_2X_3$=1011, use companion matrix to find out the next state of the LFSR.

4). This LFSR can be used as pseudo-random pattern generator for built-in self-test (BIST). For this purpose, can we start from all zero state ($X_3X_2X_1X_0$=0000)? Why?

5). Based on this LFSR design, you can revise it into a response compactor by adding a XOR gate to the D input of leftmost flip-flop (X_0). Clearly show your revision in Figure 2. After the revision, the response compactor can take 1 input for response compaction. Assume initial seed value of the LFSR as $X_3X_2X_1X_0$=0000, and output sequence of 11011001 is input to the response compactor for compaction (the rightmost bit is input to the response compactor first). Use polynomial division to find out the signature (final output states of the LFSR) after compaction.

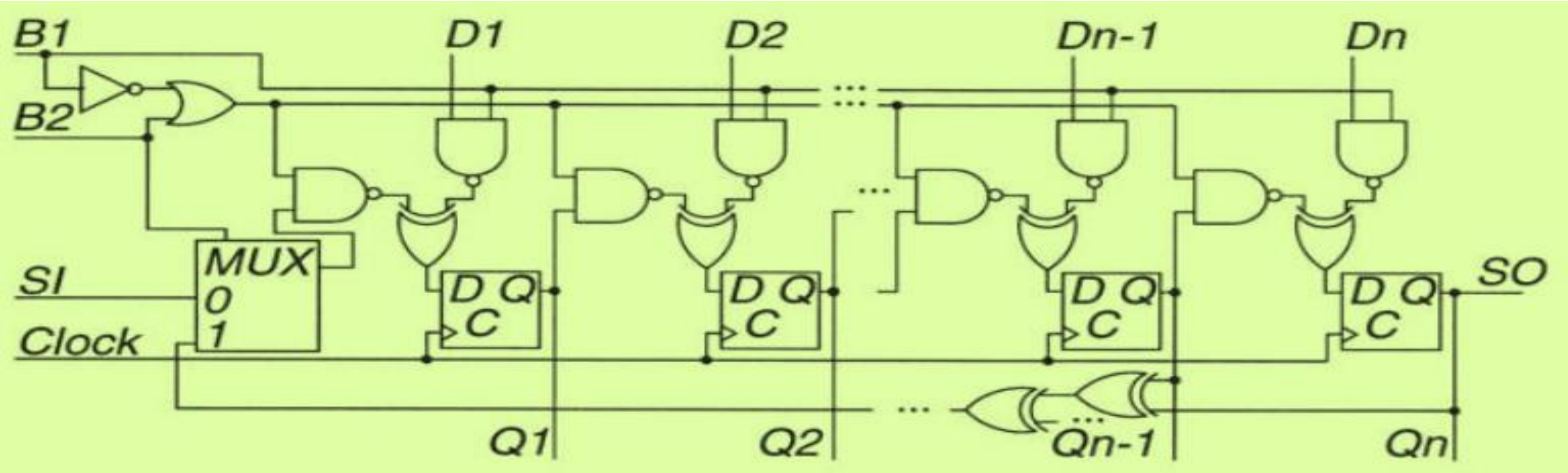
- ✓ The resulting **XOR of th** **signatures o** **polynomial d** the inputs by polynomial.
- ✓ The final signature obtained by XORing the individual signatures is 001

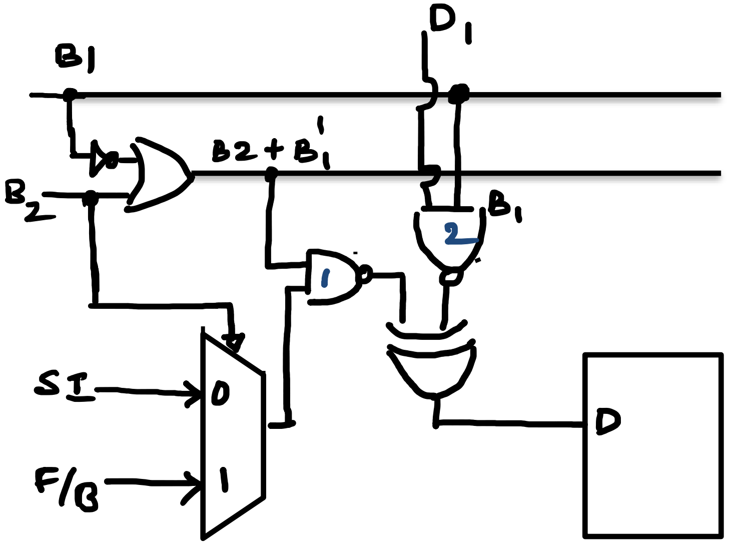
Built In Self Test (BIST)

BIST using Built-In Logic Block Observers (BILBO)

BILBO is bank od DFF with test hardware added to work in one of the Four Modes. The mode selection is based B1 and B2 inputs.

B1	B2	Mode of Operations
0	0	Serial scan chain
0	1	Standard LFSR Pattern Generator
1	0	Normal D FF
1	1	MISR Response Compactor





B2 = 1 ; F/B is selected as o/p of MUX in stage 1 & goes as i/p NAND1

– All other stages [illegible] i/p to NAND1

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

BIST using Built-In Logic Block Observers (BILBO)

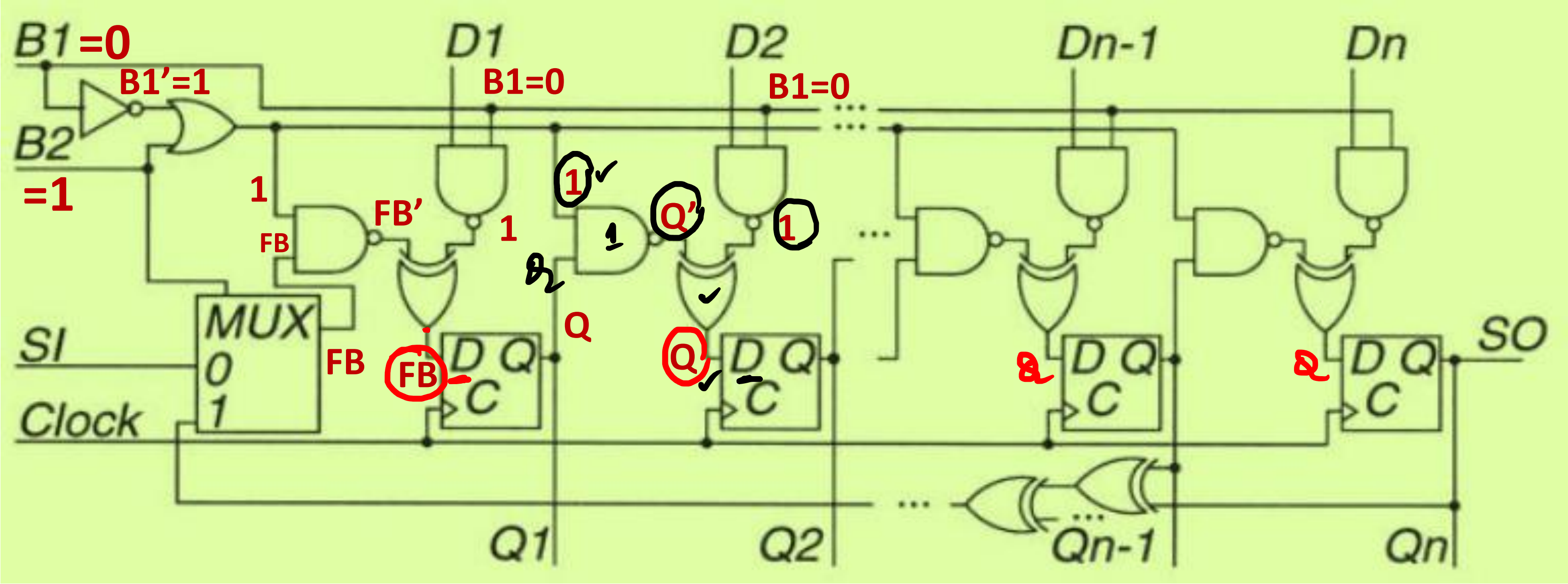
$1+x^n$

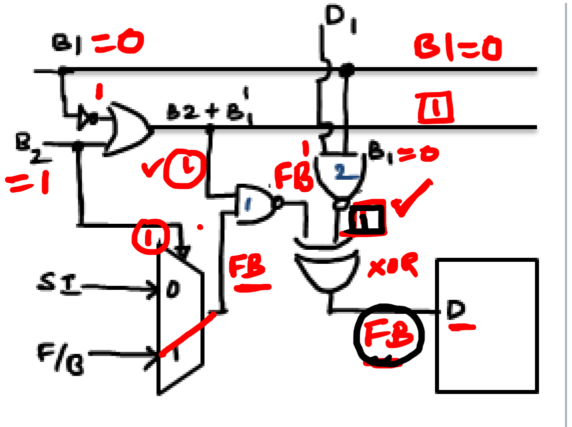
$1+x^{n-1}+x^n$

a	b	NAND	XOR	OR
0	0	1	0	0
0	1	1	1	1
1	0	1	1	1
1	1	0	0	1

BILBO is bank od DFF with test hardware added to work in one of the Four Modes. The mode selection is based B1 and B2 inputs.

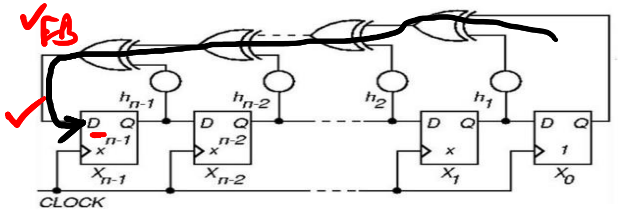
B1	B2	Mode of Operations
0	1	Standard LFSR Pattern Generator ✓





B2=1; F/B is selected as o/p of mux in stage 1 & goes as i/p NAND,
— All other stages Q & i/p to NAND1

Std LFSR used pattern gen



Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

BIST using Built-In Logic Block Observers (BILBO)

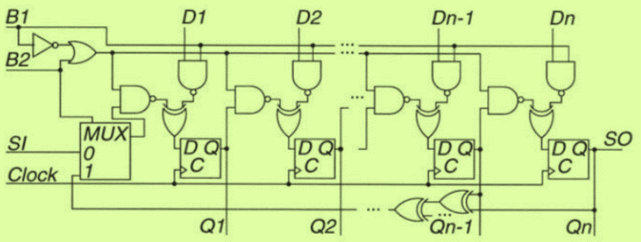
BILBO is bank od DFF with test hardware added to work in one of the Four Modes. The mode selection is based B1 and B2 inputs.

B1	B2	Mode of Operations
0	1	Standard LFSR Pattern Generator

- ✓ Since B1 = 0, all D inputs become redundant and the outputs of the upper NAND gates are all 1. Hence XOR gates invert any input coming to it.
- ✓ Since B2=1, the horizontal NAND gates pass the inverted value which is again inverted by XOR gates

- ✓ B2 also forms the select line for the MUX. B2 = 1 implies the feedback input is selected by the MUX.
- ✓ The **XOR gates in the feedback path are connected to the outputs of the D FFs depending on the characteristic polynomial** f(x)

a	b	NAND	XOR	OR
0	0	1	0	0
0	1	1	1	1
1	0	1	1	1
1	1	0	0	1

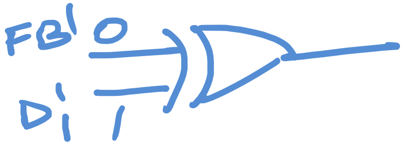


Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

BIST using Built-In Logic Block Observers (BILBO)

FB=1
D1=0

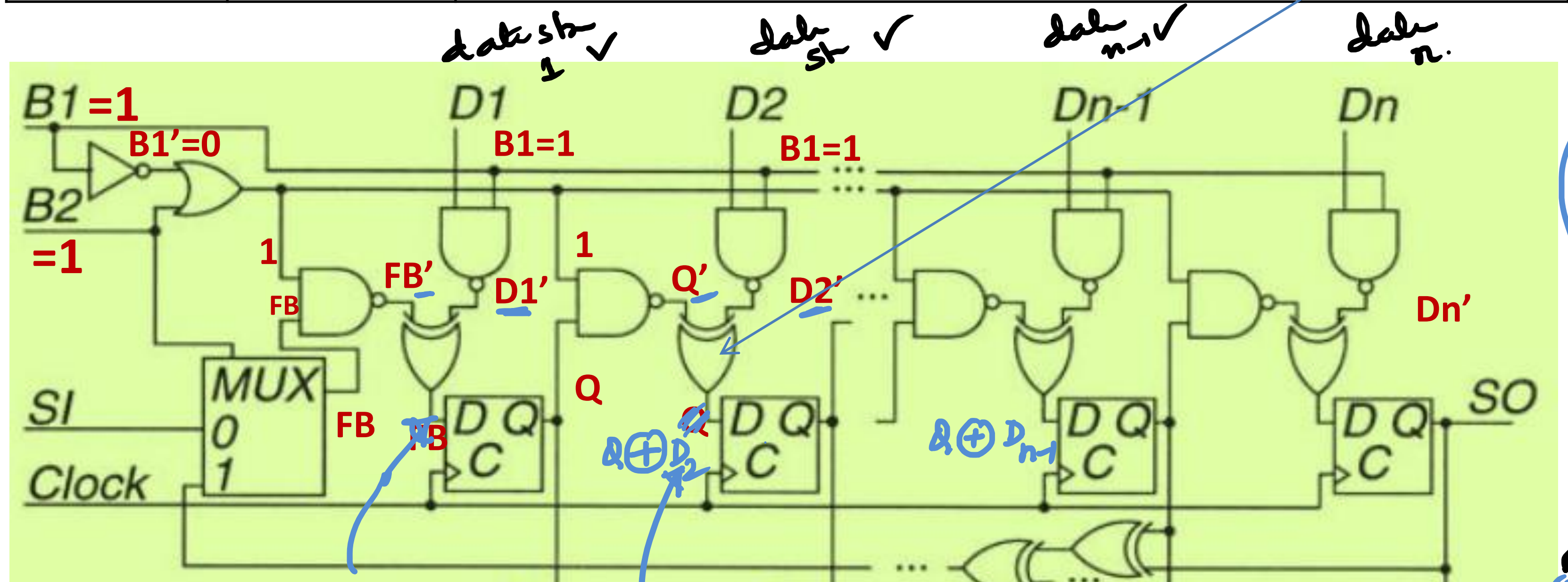


0 0 → 0
0 1 → 1
1 0 → 1 ✓
1 1 → 0

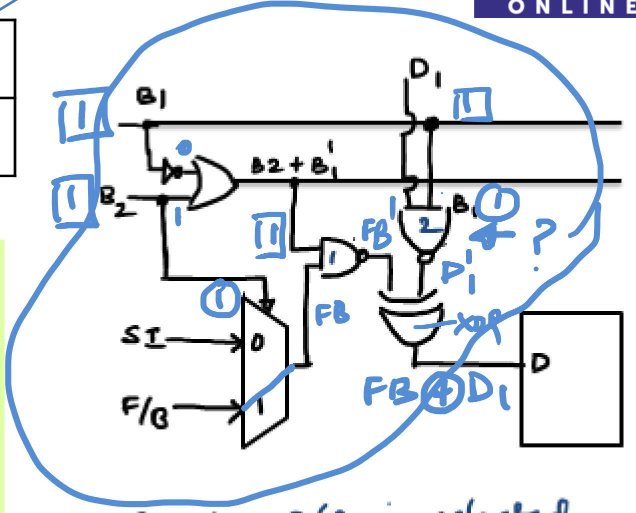
Both Inputs for XOR gate are Inverted as a result XOR o/p is NOT effected

BILBO is bank od DFF with test hardware added to work in one of the Four Modes. The mode selection is based B1 and B2 inputs.

✓ B1	B2	Mode of Operations
1 ✓	1 ✓	MISR Response Compactor (Standard LFSR as MISR)



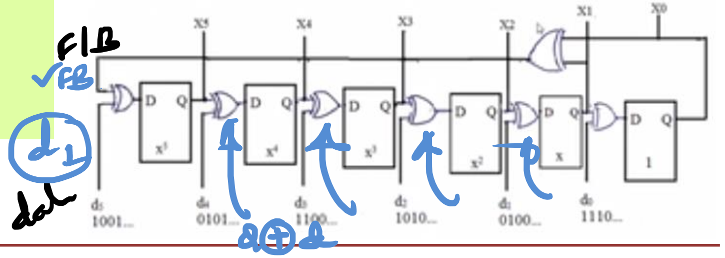
a	b	NAND	XOR	OR
0	0	1	0	0
0	1	1	1	1
1	0	1	1	1
1	1	0	0	1



B2=1 ; F/B is selected as o/p of MUX is stage 1 & goes as i/p NAND, – All other stages Q & i/p to NAND1

Q
data stream

MISR as std F



Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

BIST using Built-In Logic Block Observers (BILBO)

BILBO is bank od DFF with test hardware added to work in one of the Four Modes. The mode selection is based B1 and B2 inputs.

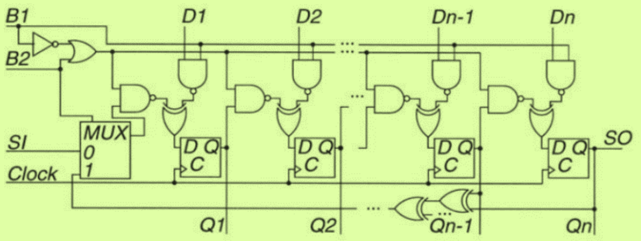
B1	B2	Mode of Operations
1	1	MISR Response Compactor (Standard LFSR as MISR)

B2 also forms the select line for the MUX. B2 = 1 implies the feedback input is selected by the MUX
The XOR gates in the feedback path are connected to the outputs of the D FFs depending on the characteristic polynomial f(x)

Since B1 = 1, the D inputs reach the XOR gates whose other input is from the output of D FF. Thus inputs of XOR gates get added at the outputs of each FF.

MISR is constructed using standard LFSR (external XOR). The 'd' inputs are the output responses of Memory at the end of each cycle

a	b	NAND	XOR	OR
0	0	1	0	0
0	1	1	1	1
1	0	1	1	1
1	1	0	0	1



Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

BIST and Redundancy

Introduction

Memories are very dense structures (denser than any logic) which allow ample opportunity for defects to impact their operation.

Memories are much denser than any logic,

- ✓ with long runs of metallization with minimum distances separating one wire from the next.
- ✓ The width of these wires is kept minimum for a given technology.
- ✓ The diffusions are packed in as tightly as possible with polysilicon, well shapes, and diffusions all at minimum distances.
- ✓ Generally, Ground rules are defined to ensure good manufacturability and printability of structures. Since many bits need to be packed as tightly as possible, ground rules are waived in the memory array region.

Defects are a part of any fabrication process and even though minute, these defects can cause havoc in a memory circuit.

A defect can be easily i) open a wire or ii) short two a) diffusions, b) polysilicon shapes, or c) metal wires together.

Failing cells are "repaired" or "replaced" !!!!!!!!!!

When a memory fails, it usually has a

- **bad single cell,** ✓
- **bad row,** ✓
- **bad column,** or ✓
- **bad cluster.** ✓

The dominant fail mode is

- ✓ single cell fails,
- ✓ followed by paired cell failures, and
- ✓ then by row or column failures .

Memories, after test, can be good, bad, or fixable(Not repaired but replaced).

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

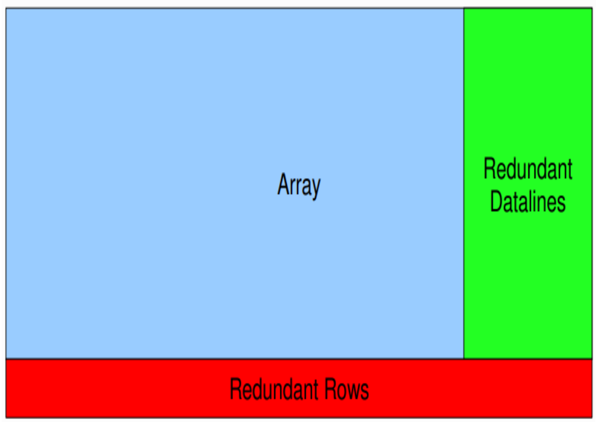
Built In Self Test (BIST)

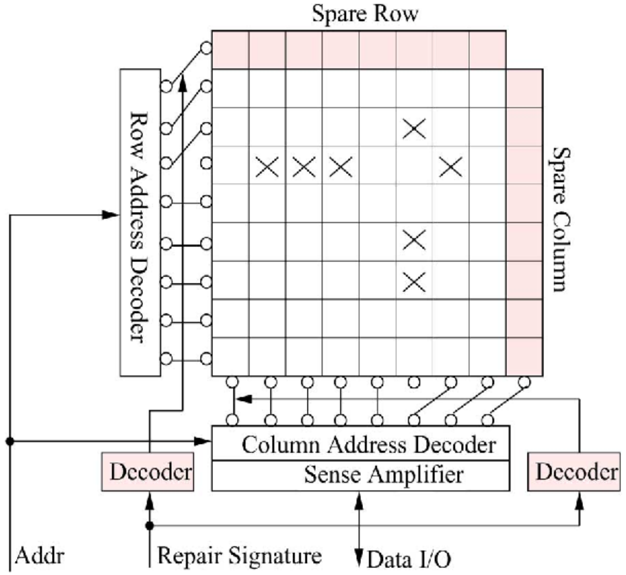
BIST and Redundancy

What is Redundancy in Memory Architecture ?

Failing cells are not "repaired" but replaced

- Memories must be able to exist in in the presence of defects. Redundancy enables this to happen.
- A Memory has extra
  - ✓ Rows,
  - ✓ Columns
  - ✓ Data lines or
  - ✓ blocks which are redundant.
- **When a defective element is detected, that element is ignored and is replaced by one of the spare elements.** i.e., the bad piece of the memory is steered around and a substitute piece is used in its place





Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

BIST and Redundancy

Redundancy types

The Memory redundancy can be Classified as

1. Single Cell redundancy
2. Row redundancy
3. Column redundancy

1. Single Cell Redundancy :

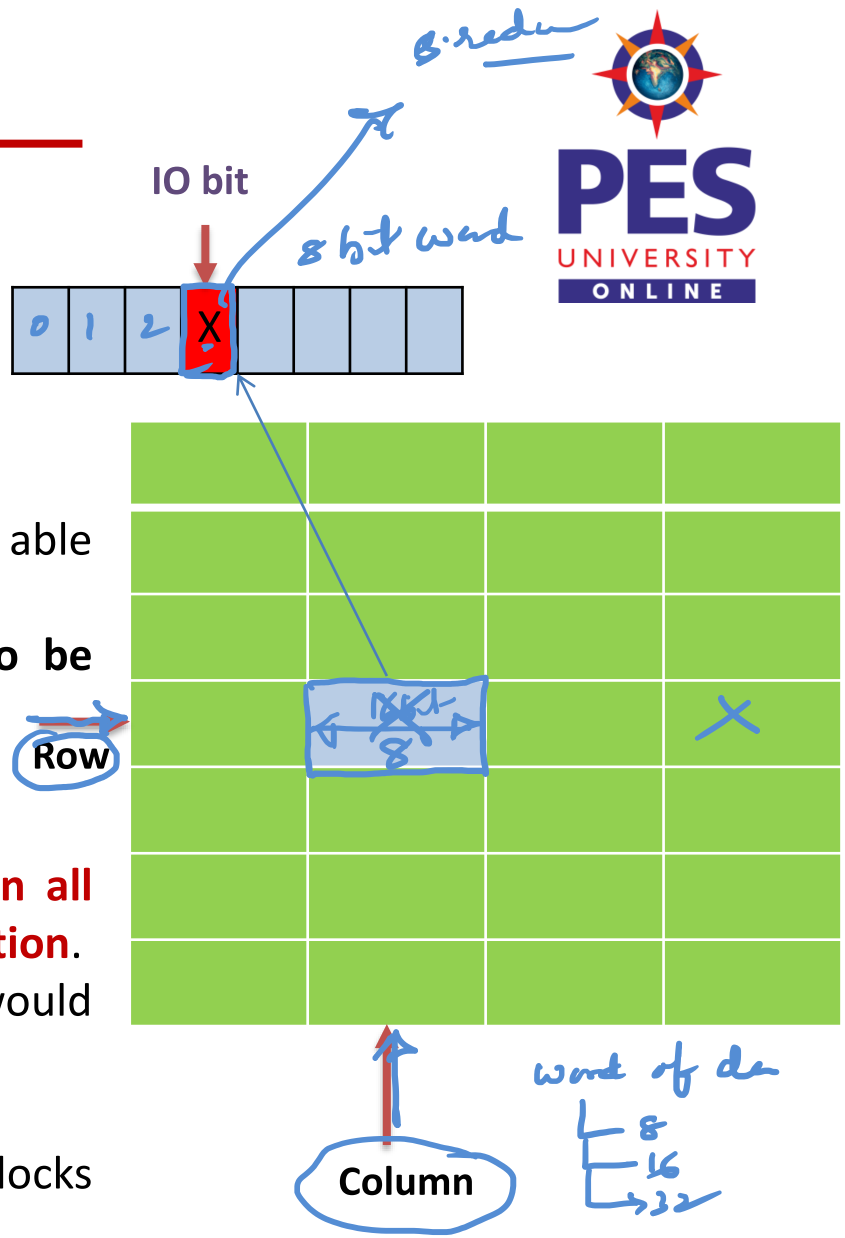
✓ Most fails on memories are single cell fails, it would be fine to be able to repair a single cell at a time.

✓ The **overhead to repair a single cell at a time is too large to be practical**.

✓ Identifying a single cell to be replaced means

- **Storing the failing row address, column address, and IO bit.**
- **On each read a compare would have to be performed on all three portions of this address against the stored failing location.**
- If the failing location is accessed, the redundant single cell would then be supplied in place of its failing counterpart.
- The time it takes for this process is too large

Instead of individual cells being replaced, rows, columns or even whole blocks or sub-arrays are replaced.



Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

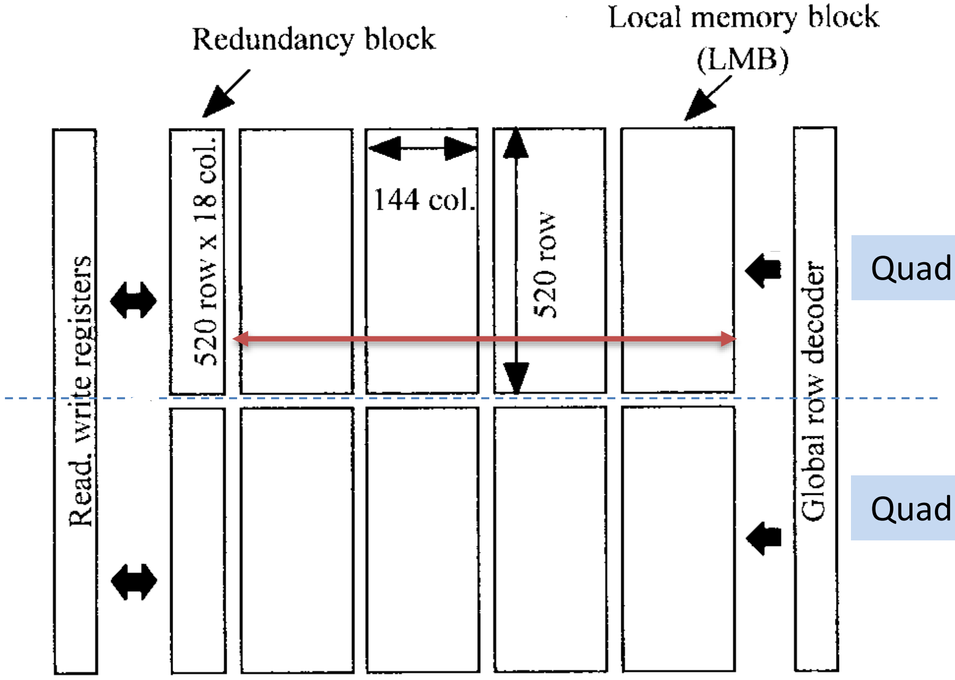
Built In Self Test (BIST)

BIST and Redundancy

Row Redundancy

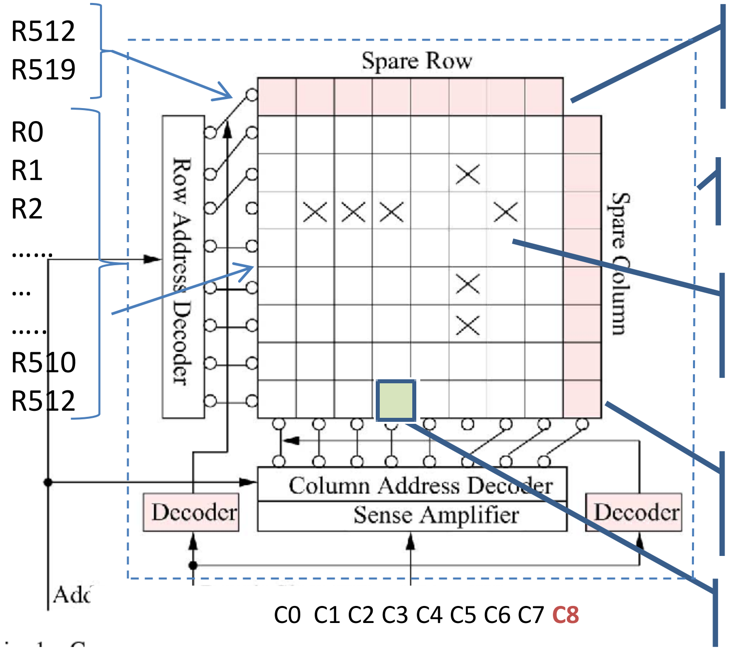
1. Redundant rows are allocated on a quadrant basis
2. Each quadrant has sub-arrays
3. When a row is replaced, it spans all columns and sub-arrays in a quadrant
4. Multiple single cell fails in 1-dimension can therefore be replaced

520-512 = 8 Rows are redundant
144-128 = 16 Extra Columns
144/16 = 9 words (8+1 spare Row for every word)



Quadrant

Quadrant



Extra Rows
Example – 8 Rows

Sub Array of 512x8x16 bits

Total Number of Column =8
Columns of 16 bit data

Extra Columns
Example – 8 Rows

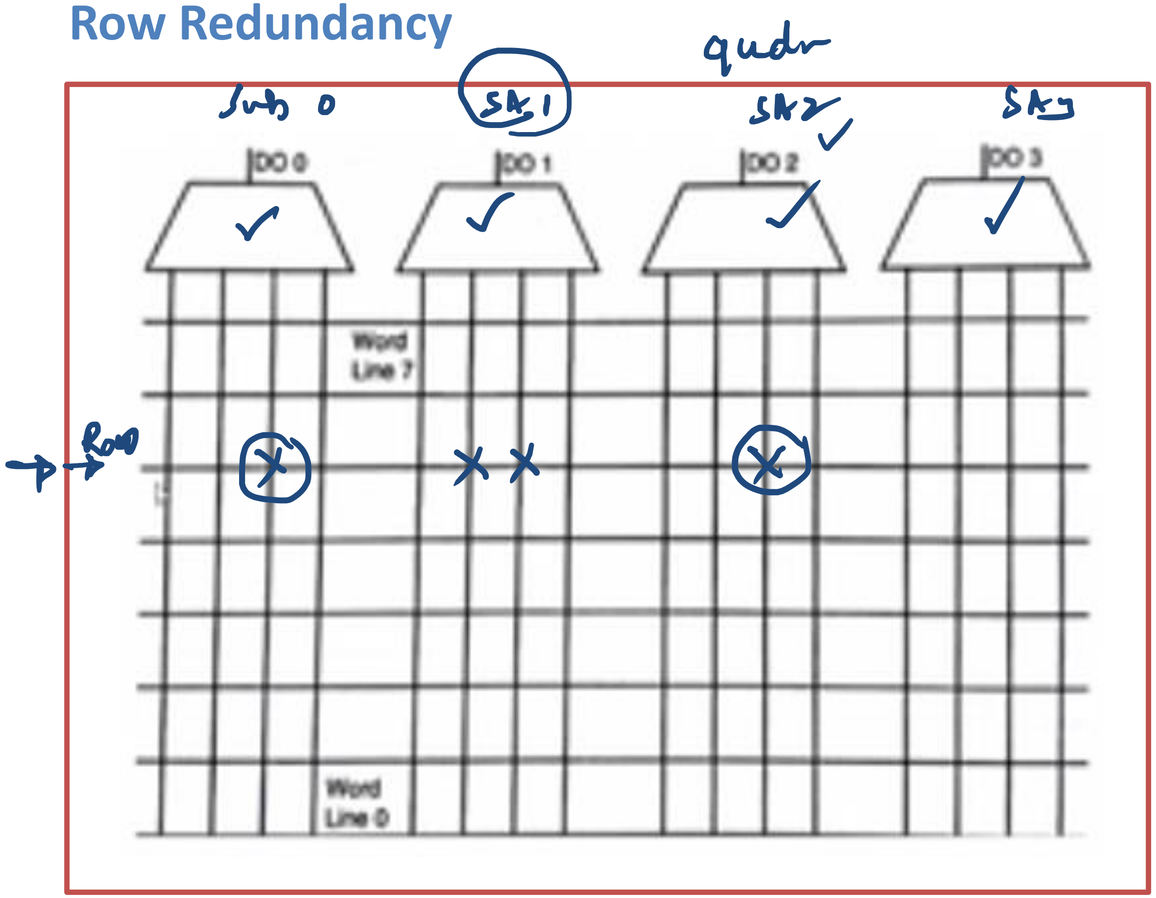
Each Column of 16 bit word

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

BIST and Redundancy

Row Redundancy



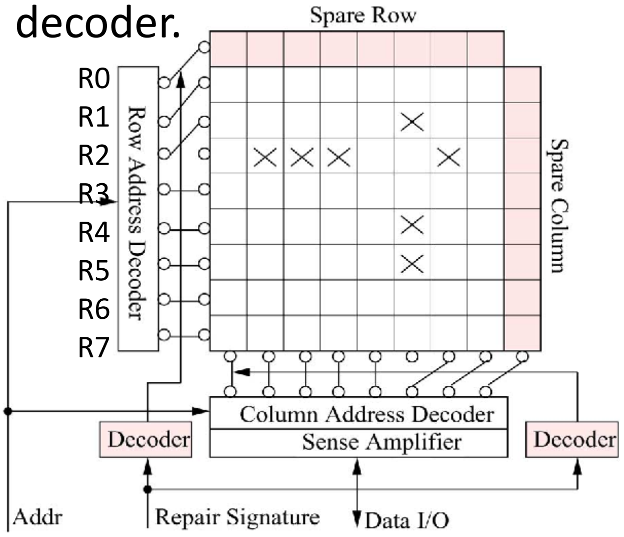
Reference :“High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

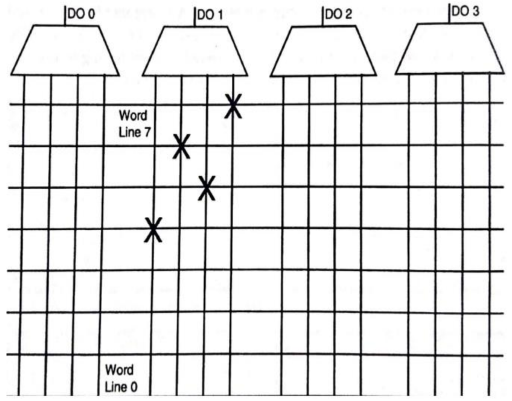
BIST and Redundancy

Column Redundancy

1. Involves **replacing a column throughout a sub-array.**
2. Along Column replacement it involves replacement of Column peripherals such as **Bit-line pair, pre-charge, isolation, sense amplifiers and write-drivers.**
3. The **overhead on replacing columns is large in Comparison with the Row replacement a**s this involve additional peripheral circuits replacement.
4. Hence entire I/O is replaced.
5. The I/O may be used to access 8, 16 or more columns depending on the column decoder.



- Let us consider Array of 8 Rows and 8 Columns .
- In this, If faults are identified row wise, then R1,R2,R4 and R5 are have Faulty Cells
- Column wise Column C5 is the only Column Faulty.
- Replacing Column C5 in the sub array will address the issue



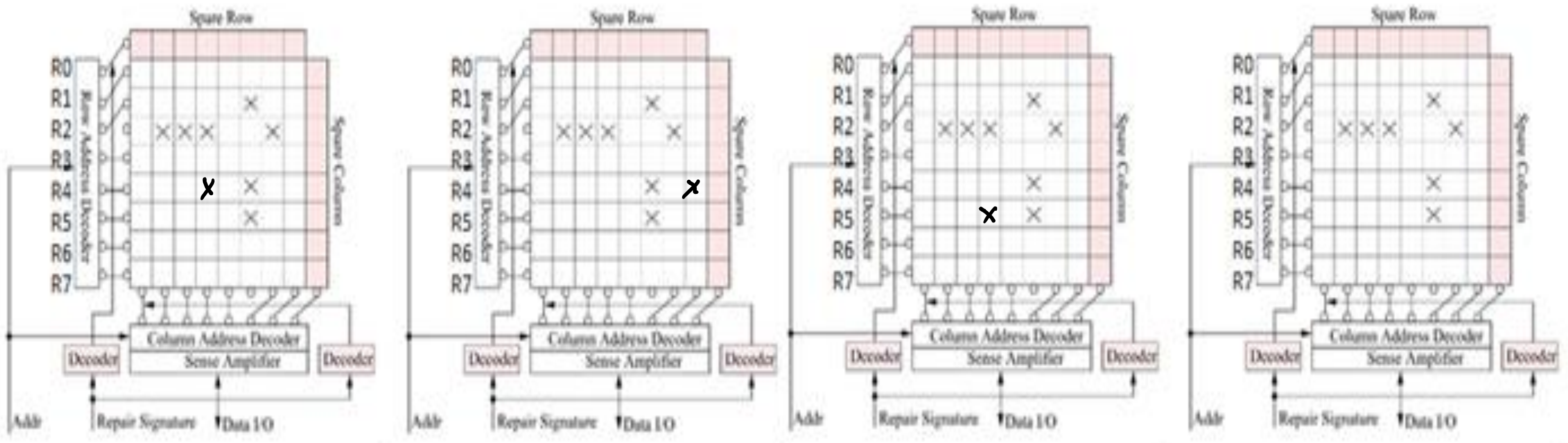
Note: When a row is replaced, it spans all columns and sub-arrays in a quadrant.

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

BIST and Redundancy

Let us consider a quadrant



Note: When a row is replaced, it spans all columns and sub-arrays in a quadrant.

- Let us consider Array of 8 Rows and 8 Columns .
- In this, If faults are **identified row wise, then R1,R2,R4 and R5** are have Faulty Cells in whole quadrant
- Column wise **Column C5 in in each sub arrays is faulty.** Therefore 4 Columns need to be replaced.
- Row wise **replacement spans all columns and sub arrays** in a quadrant and **Replacing 4 rows addresses the C5 faults also**. It is preferred to replace Rows

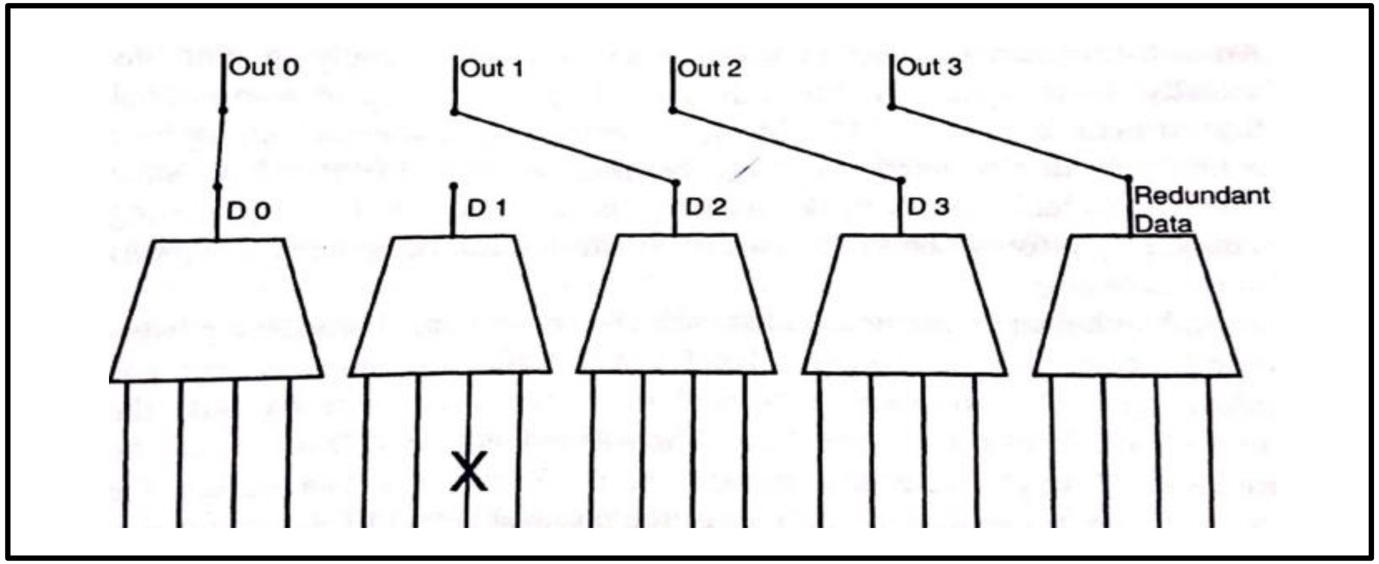
Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

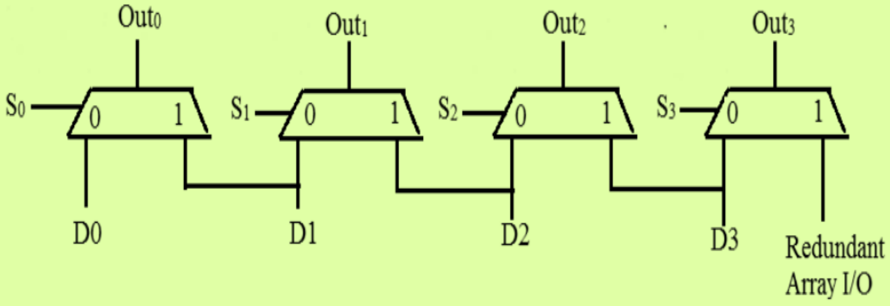
BIST and Redundancy

MUX Arrangement

- Replacement of Sub-Array 1 and its peripheral circuitry with Redundant array .
- Replacing the I/O corrects non-cell defects as well, such as bit-line shorts, pre-charge defects, write-driver defects etc.



MUX arrangement to do the Column replacement



Ex: Sub- Array 1 Replacement
S0S1S2S3 =0111
Ex: Sub- Array 3 Replacement
S0S1S2S3 =0001

Reference :“High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

BIST and Redundancy

Redundancy Challenges

- Amount of redundancy on a memory is dependent on the type and size of the memory and the silicon area that can be allocated to redundant elements.
- Redundancy for CAM, specially TCAM which stores 3 states, is challenging since the compare circuitry needs to be replaced along with the memory cells.
- **1st type of redundancy implemented is Row.**
- **Next depending on the defect sites, either entire sub-array is replaced or a single column.**

✓ Redun Row & Colm

Do redundant Memory Faulty ?

- A redundant memory element may have defects.
- In such cases, a failing memory element may be replaced by another failing element
- Hence redundant elements also need to be tested before replacement

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

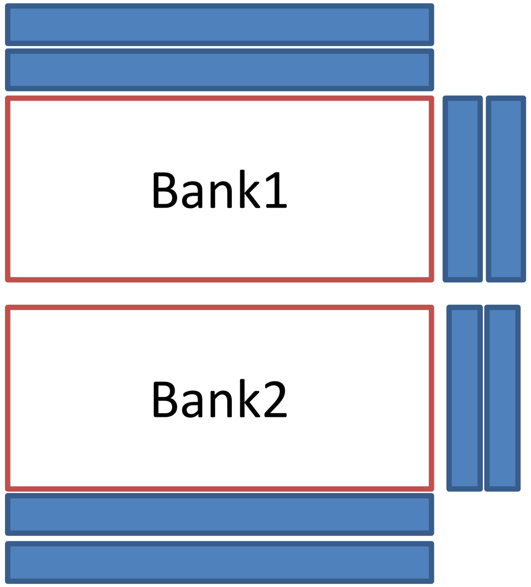
Built In Self Test (BIST)

BIST and Redundancy

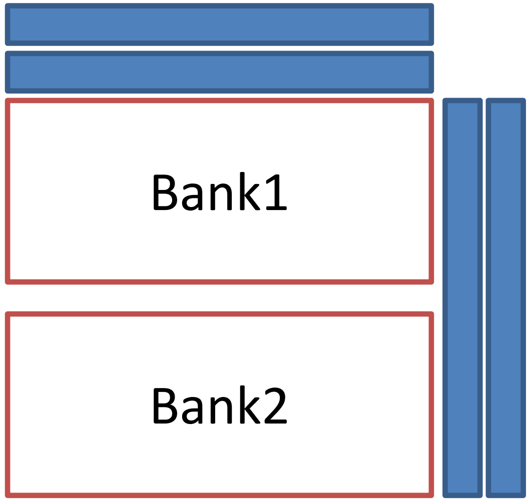
Redundancy Organization

There are 3 types of Redundancy Organization

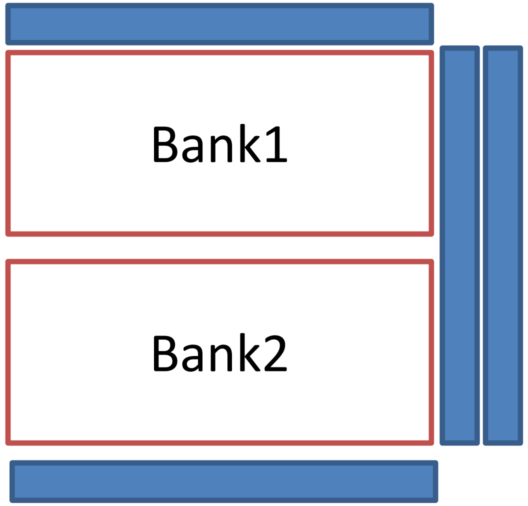
1. Local redundancies : Redundant rows and columns are local to each memory bank
2. Global redundancies :Redundant rows and columns are Global to each memory bank
3. Hybrid redundancies : Combination of local and global redundancies



Local redundancies



Global redundancies



Hybrid redundancies

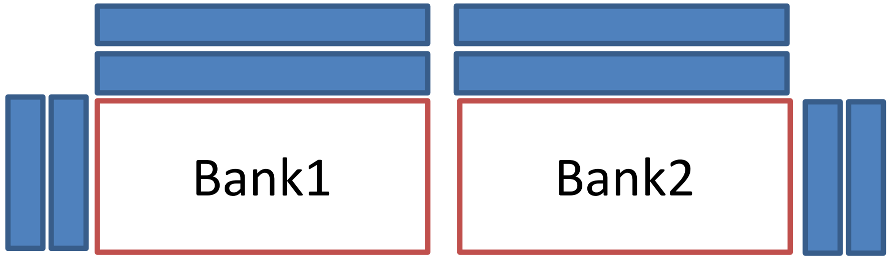
Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

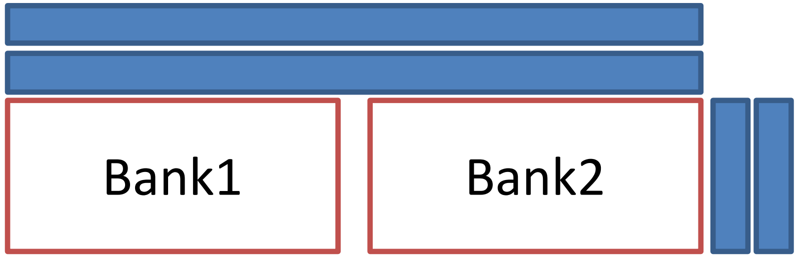
BIST and Redundancy

Redundancy Organization

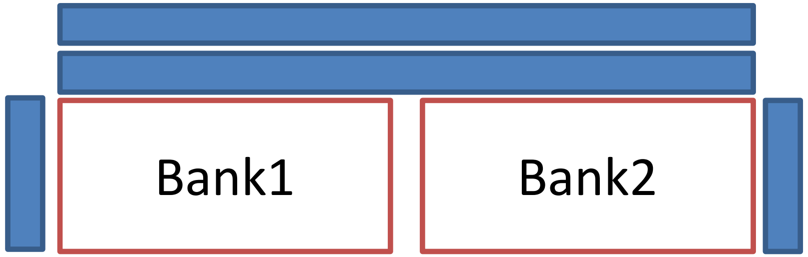
Local redundancies



Global redundancies



Hybrid redundancies



There are 3 types of Redundancy Organization

1. Local redundancies : Redundant rows and columns are local to each memory bank

2. Global redundancies :Redundant rows and columns are Global to each memory bank

3. Hybrid redundancies : Combination of local and global redundancies

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

BIST and Redundancy

The Redundancy Calculation

Built In Self Repair (BISR)

- BISR provides the logic for the redundancy calculation and the replacement procedure.
- BISR logic employs 2-pass testing of memory to calculate redundancy
  - ✓ **In the 1st Pass**: **Number of fails along rows and columns is counted**
  - ✓ On exceeding the specified number along a particular dimension, a **"MUST FIX" row** or column is defined and marked for repair

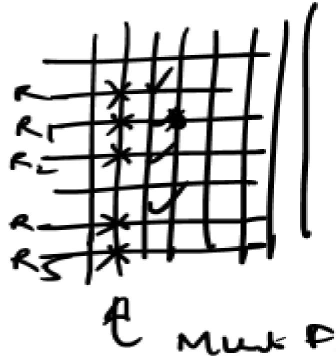
  Ex: **if there are four redundant rows and five fails are detected along a column, a must-fix column selection can be made.**
  - ✓ **In the 2nd Pass:** Test is performed to allocate the remaining redundancies to spare failures
  - ✓ Even the best BISR implementation cannot assure 100% repair

4 Redund Row, Redun col

5 faults → 5 differ

4 rows.

BISR Unit



Must Fix col

Even with a good redundancy calculation method, the best BIST implementation also cannot assure a repair of 100% of the fixable memories. Proper BIST implementation of a redundancy calculation, though, can provide significant yield enhancement and can also assure that no failing memories are shipped to the field.

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

BIST and Redundancy

The Redundancy Calculation

Built In Self Repair (BISR)

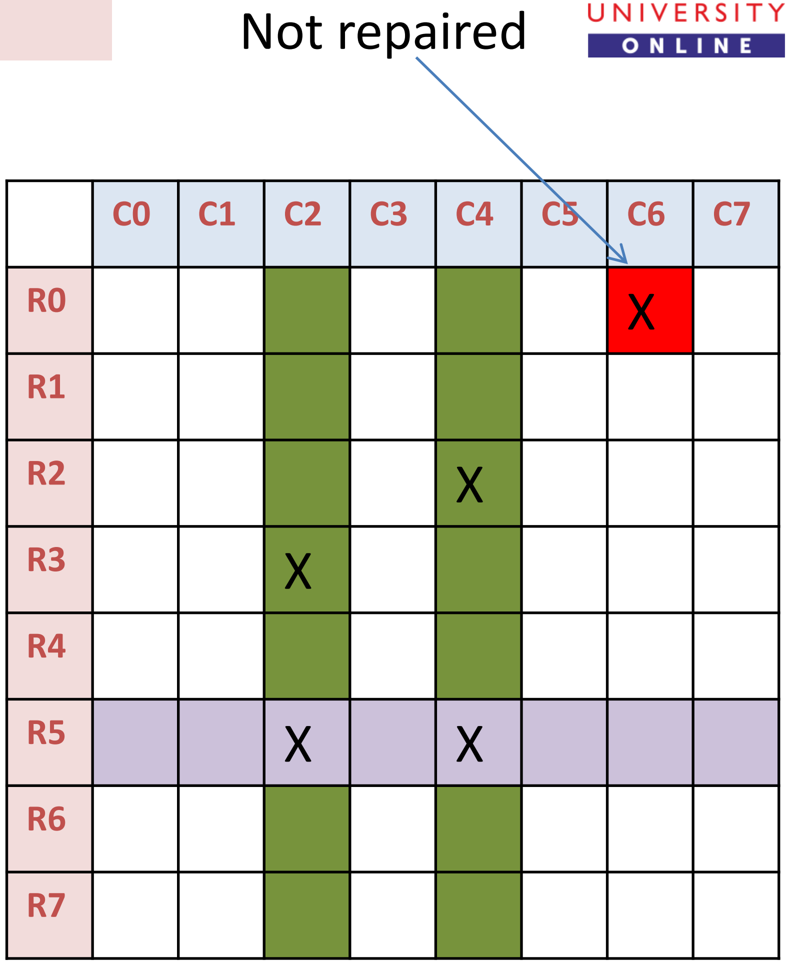
Consider an 8 x 8 memory array with 5 failing cells

	C0	C1	C2	C3	C4	C5	C6	C7
R0							X	
R1								
R2					X			
R3			X					
R4								
R5			X		X			
R6								
R7								

Number of Fails along the Rows = 5
R0 -1F, R2-1F , R3-1F, R5-2F
Number of Fails along the Column= 5
C2-2F, C4-2F, C6-1F

Ex: **If Redundant Rows and Column available are 1 and 2 resp. .**

- It would be **easy to allocate a redundant row to the pair of fails in row 5.**
- The 2 redundant columns replace C2 and C4. **This means Fault in R0C6 is NOT covered by ANY redundant elements**



Not repaired

	C0	C1	C2	C3	C4	C5	C6	C7
R0							X	
R1								
R2					X			
R3			X					
R4								
R5			X		X			
R6								
R7								

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

BIST and Redundancy

The Redundancy Calculation

Built In Self Repair (BISR)

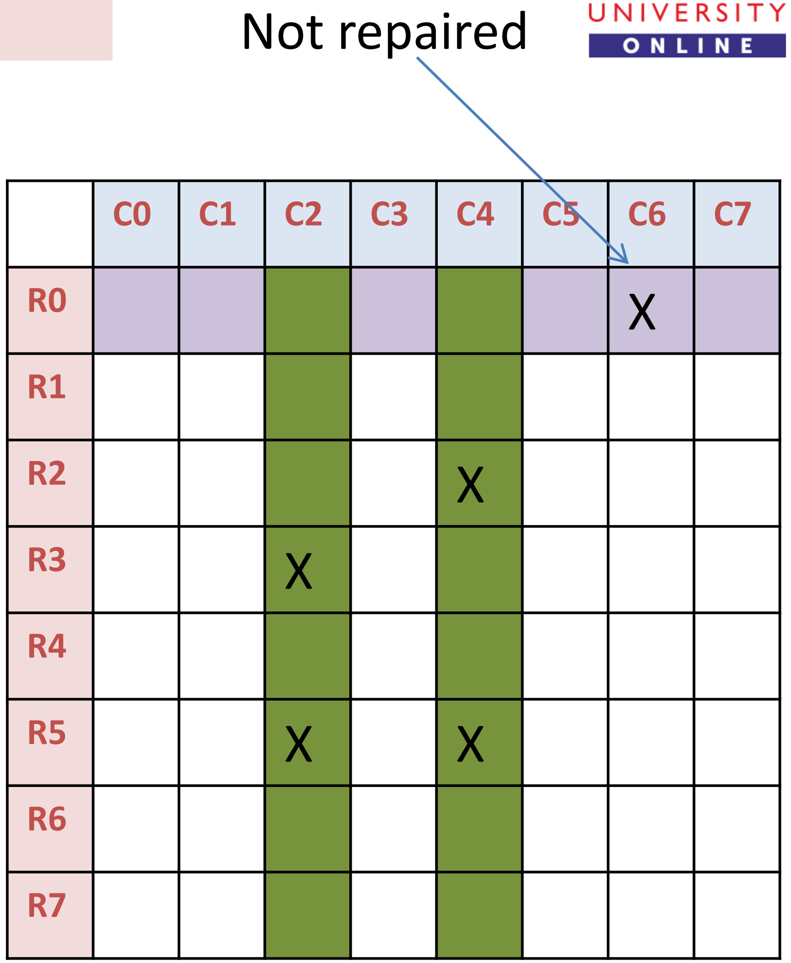
Consider an 8 x 8 memory array with 5 failing cells

	C0	C1	C2	C3	C4	C5	C6	C7
R0							X	
R1								
R2					X			
R3			X					
R4								
R5			X		X			
R6								
R7								

Number of Fails along the Rows = 5
R0 -1F, R2-1F , R3-1F, R5-2F
Number of Fails along the Column= 5
C2-2F, C4-2F, C6-1F

Ex: **If Redundant Rows and Column available are 1 and 2 resp.** .

- If the fails in columns 2 and 4 are replaced with redundant columns, the fail in row 0 can be replaced by the redundant row, allowing full repair.



Not repaired

	C0	C1	C2	C3	C4	C5	C6	C7
R0							X	
R1								
R2					X			
R3			X					
R4								
R5			X		X			
R6								
R7								

Reference :“High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

BIST and Redundancy

The Redundancy Calculation

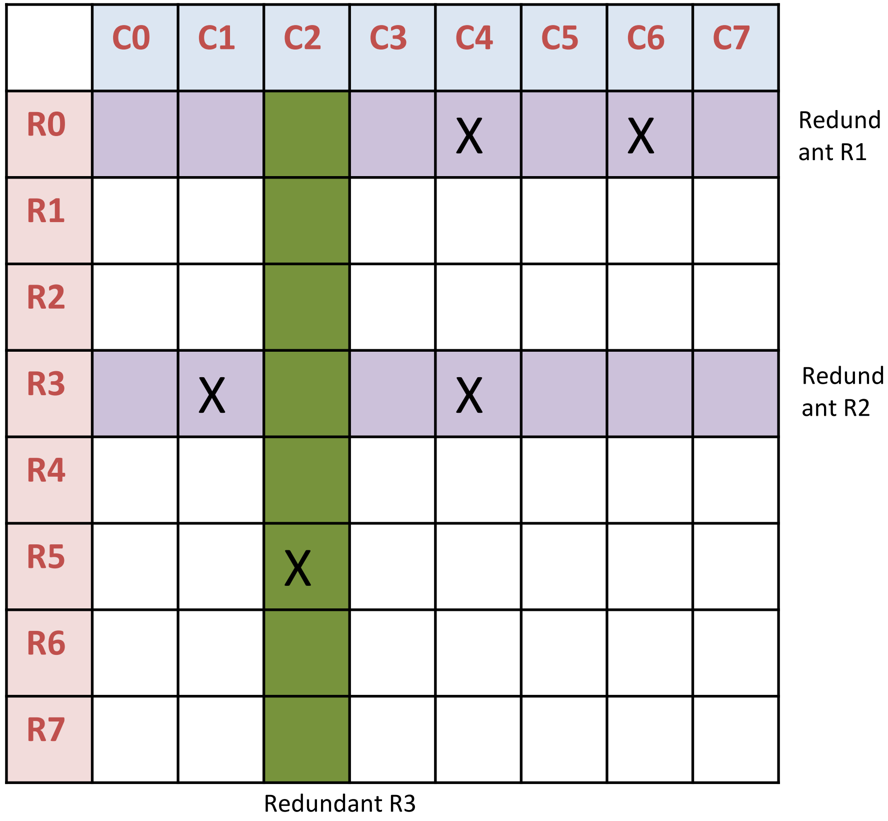
Built In Self Repair (BISR)

Consider an 8 x 8 memory array with 5 failing cells

	C0	C1	C2	C3	C4	C5	C6	C7
R0					X		X	
R1								
R2								
R3		X			X			
R4								
R5			X					
R6								
R7								

Number of Fails along the Rows = 5
R0 -2F , R3-2F , R5-1F
Number of Fails along the Column= 5
C1-1F ,C2-1F , C4-2F ,C6-1

Ex: If there are Redundant Rows and Column available are 2 each .
BISR will replace R0 and R3 by 2 available redundant Rows and C2 by a redundant Columns



	C0	C1	C2	C3	C4	C5	C6	C7
R0					X		X	
R1								
R2								
R3		X			X			
R4								
R5			X					
R6								
R7								

Redundant R1

Redundant R2

Redundant R3

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

BIST and Redundancy

The Redundancy Calculation

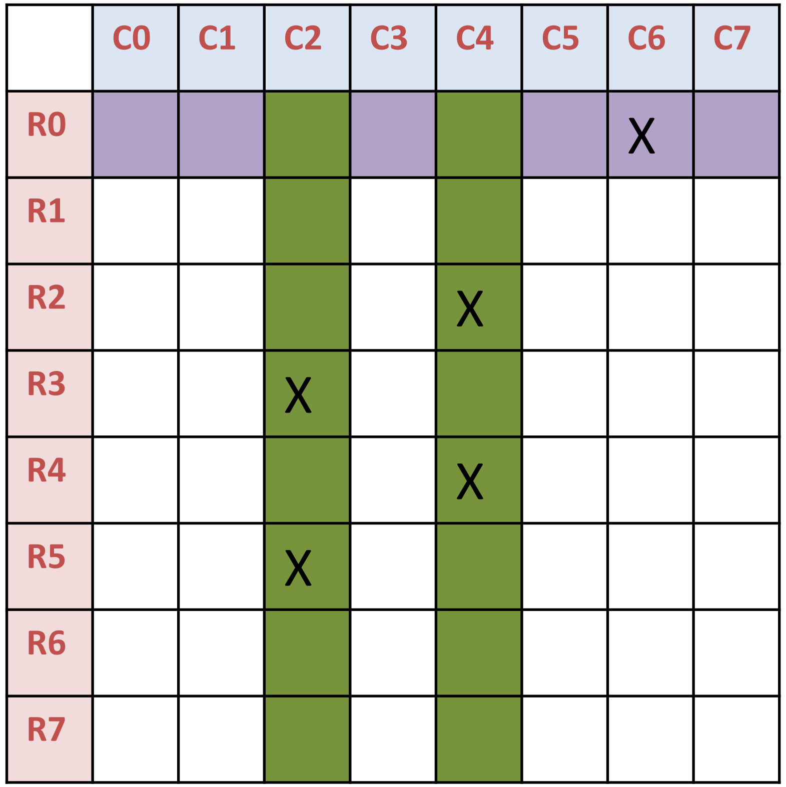
Built In Self Repair (BISR)

Consider an 8 x 8 memory array with 5 failing cells

	C0	C1	C2	C3	C4	C5	C6	C7
R0							X	
R1								
R2					X			
R3			X					
R4					X			
R5			X					
R6								
R7								

Number of Fails along the Rows = 5
R0 -1F, R2-1F , R3-1F, R4-1F,R5-1F
Number of Fails along the Column= 5
C2-2F, C4-2F, C6-1F

Ex: **If Redundant Rows and Column available are 3 and 2 resp.**
BISR will replace R0 by redundant Row and C2 and C4 by redundant Columns



	C0	C1	C2	C3	C4	C5	C6	C7
R0							X	
R1								
R2					X			
R3			X					
R4					X			
R5			X					
R6								
R7								

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

BIST and Redundancy

The Redundancy Calculation

Built In Self Repair (BISR)

Consider an 8 x 8 memory array with 5 failing cells

	C0	C1	C2	C3	C4	C5	C6	C7
R0					X		X	
R1								
R2								
R3		X			X			
R4								
R5			X					
R6								
R7								

Number of Fails along the Rows = 5
R0 -2F , R3-2F , R5-1F
Number of Fails along the Column= 5
C1-1F ,C2-1F , C4-2F ,C6-1

Ex: If Redundant Rows and Column available are 3 and 2 resp.
BISR will replace R0 and R3 by 2 available redundant Rows and C2 by a redundant Columns

	C0	C1	C2	C3	C4	C5	C6	C7	
R0					X		X		Redundant R1
R1									
R2									
R3		X			X				Redundant R2
R4									
R5			X						Redundant R3
R6									
R7									

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Memory Error Detection and Correction Techniques

Types of Errors

Errors in Semiconductor Memories can be broadly classified into 2 types

1. **Hard Errors:** Faults which are permanent
   - Stuck faults and permanent physical damage to the memory
   - Such faults can only be detected but not corrected
   - Redundancy techniques can be used to "repair" such memory
2. **Soft Errors: They only corrupt the value stored** in the memory cell **but do not permanently damage the hardware.**
   - Errors due to **external radiation or electrical noise** rather than design or manufacturing defects
   - Caused by **high energy electrons, alpha particles and cosmic rays** hitting the Silicon bulk
   - Result in the **production of large number of electron-hole pairs leading to parasitic leakage current**
   - The **accumulated charge may be sufficient to flip** the value stored in the cell.

   All these may hence cause bit inversion of a storage cell, resulting in soft error

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

Memory Error Detection and Correction Techniques

Types Soft Errors:

1. **Single Event Upset (SEU) :** A single ionizing particle affecting a single bit is an SEU

2. **Multiple Bit Upset (MBU)**: An event or series of events that **cause more than one bit to be upset during a single measurement.** The probability of adjacent double bit errors is much higher than other multiple bit errors

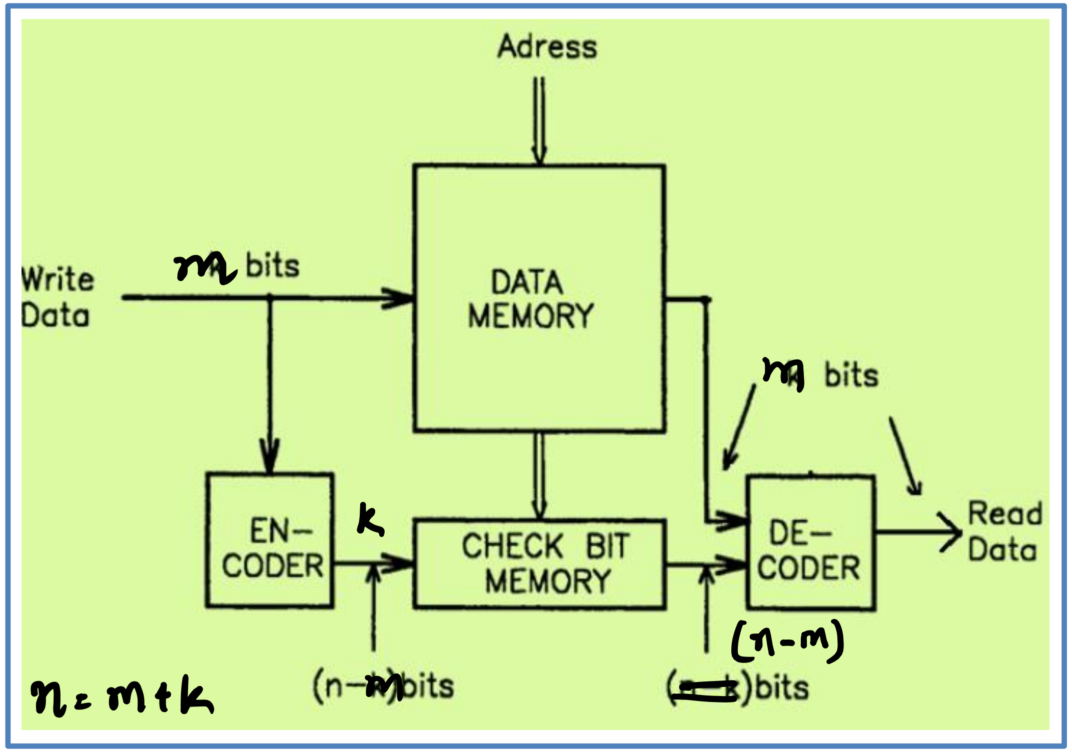
Built In Self Test (BIST)

Error Correction Codes (ECC)

- ECC is the Most common approach to protect the memories against the soft errors.
- ECC can be applied to a data word of any length

Algorithm for testing memory using ECC

1) During a **memory write operation the data is encoded in an encoder, using the ECC to generate the check bits** (redundant bits). The check bits are stored in a separate memory space.
2) During a memory read operation, **the check bits are retrieved along with the data bits into a decoder.**
3) The decoder uses the **check bits to correct any error in the data bits** and the **corrected data is sent out through the data bus**



Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

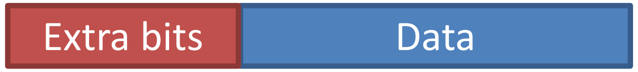
Built In Self Test (BIST)

Memory Error Detection and Correction Techniques

Error detection methods used for Soft Errors

Basic Concept:

- Extra bits (check bits)are added to the data bit that we want to represent , to get **code words.**



- The **check bits** help in **detecting the soft error** in the data i.e., Data is correct or Faulty.
- The **code words** are divided into two categories as
  - ✓ **Valid code word**
  - ✓ **Invalid code word**
- A **bit error changes a Valid code word into a Invalid code word**

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

Memory Error Detection and Correction Techniques

Error detection methods used for Soft Errors

Parity Check scheme :It is the Simplest approach which can be used for Error Detection

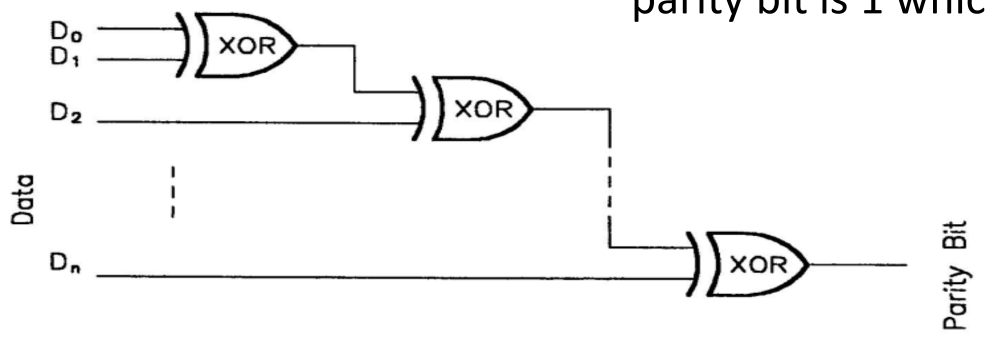
- Parity of a given data is defined by whether the number of 1's in the data is Odd or Even. If it is Odd number of 1s then it is called Odd Parity else it is Even Parity
- Add an **extra parity bit 'p' in the MSB to make the number of 1's in each code word even**
- A single error changes the valid code word into an invalid one which can be detected

Extra bits	Data

Example: Even parity BCD Code

P	d2	d1	d0	
0	0	0	0	
1	0	0	1	
1	0	1	0	
0	0	1	1	
1	1	0	0	
0	1	0	1	
0	1	1	0	
1	1	1	1	

Parity bit generation circuit



If data has odd parity, then the generated parity bit is 1 which is added to the MSB

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

Memory Error Detection and Correction Techniques

Error detection methods used for Soft Errors

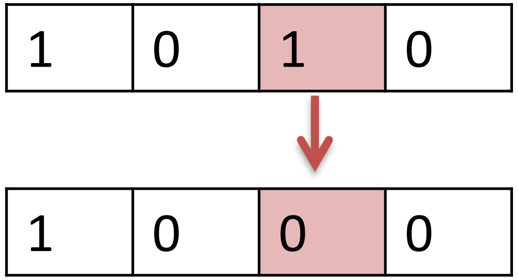
Parity Check scheme :

- **It is the** Simplest approach which can be used for Error Detection
- A single error changes the valid code word into an invalid one which can be detected

P	d2	d1	d0	CW	Parity of CW
0	0	0	0	Valid(0)	0 (Even)
1	0	0	1	Valid(9)	0 (Even)
1	0	1	0	Valid(A)	0 (Even)
0	0	1	1	Valid(3)	0 (Even)
1	1	0	0	Valid(C)	0 (Even)
0	1	0	1	Valid(5)	0 (Even)
0	1	1	0	Valid(6)	0 (Even)
1	1	1	1	Valid(F)	0 (Even)

P	d2	d1	d0	CW
0	0	0	1	Invalid(1)
0	0	1	0	Invalid(2)
0	1	0	0	Invalid(4)
0	1	1	1	Invalid(7)
1	0	0	0	Invalid(8)
1	0	1	1	Invalid(B)
1	1	0	1	Invalid(D)
1	1	1	0	Invalid(E)

Example: Even parity BCD Code



1	0	1	0

1	0	0	0

A single error changes in Valid code word leads to Invalid code word

Reference :“High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

Memory Error Detection and Correction Techniques

Error detection methods used for Soft Errors

Parity Check scheme : It is the Simplest approach which can be used for Error Detection

Hamming Distance and Parity Code

- Hamming distance is the number of bit positions by which code-word differs.
- If can be obtained by Counting number of 1s in bitwise XOR of two code-words

cw1	1	0	0	1
cw2	0	1	0	1
	1	1	0	0

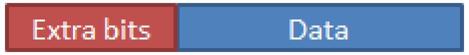
Distance (d)=2

- ✓ To detect 'b' bit errors, the Minimum distance i.e., **dmin=b +1**
- ✓ To correct 'b' bit errors, Minimum distance **i.e., dmin=2b+1**

Ex: If dmin=2 ;

a) Number of Errors Detected is b= dmin-1 =2-1=1 . It means **One error can be detected.**

b) Errors corrected is
b= (dmin-1)/2 = (2-1)/2 =0.5
It means **No Error can be corrected**



Example: Even parity BCD Code

P	d2	d1	d0	
0	0	0	0	
1	0	0	1	
1	0	1	0	
0	0	1	1	
1	1	0	0	
0	1	0	1	
0	1	1	0	
1	1	1	1	

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

Memory Error Detection and Correction Techniques

Error detection methods used for Soft Errors

Hamming Code for Error Detection and Correction

- Hamming Code can be used for **Single Error Correction**
- **It uses More than 1 Parity bits. i.e., Let us say k parity bits are added to form a (m+k) bits data word.**
- **k parity bits must satisfy In equality $2^k \geq (m+k+1)$**
- **Data is of m bit size**
- Parity bits in the code word are **positioned at 2^{ith} position and Rest of the bits are filled by data bits.**
- 2^{ith} position **are $2^0, 2^1, 2^2, 2^3$…….**
- The parity check bits are computed based on some well defined formulas

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2^0	2^1		2^2				2^3								2^4
P1	P2	D3	P4	D5	D6	D7	P8	D9	D10	D11	D12	D13	D14	D15	P16

k=3

k parity bits must satisfy In equality $2^k \geq (m+k+1)$

Ex: m=4, k=?

k=2 ; $2^2 \geq (4+2+1)$; $4 \not\geq 7$

k=3 ; $2^3 \geq (4+3+1)$; $8 \geq 8$

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

Memory Error Detection and Correction Techniques

Error detection methods used for Soft Errors

Hamming Code for Error Detection and Correction

Hamming Code Generation:

Example (7,4) Code: Rate =4/7

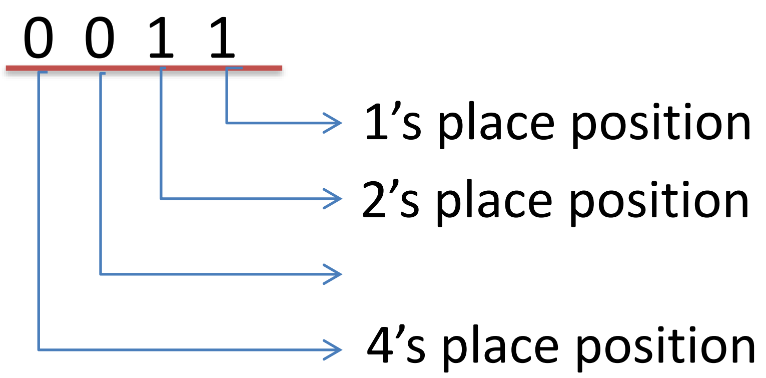
Binary Coded Positions →

0001	0010	0011	0100	0101	0110	0111
2^0	2^1		2^2			
P1	P2	D3	P4	D5	D6	D7

$P_1 = D_3 \oplus D_5 \oplus D_7$: All with 1 in 1's place of position

$P_2 = D_3 \oplus D_6 \oplus D_7$: All with 1 in 2's place of position

$P_4 = D_5 \oplus D_6 \oplus D_7$: All with 1 in 4's place of position



Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

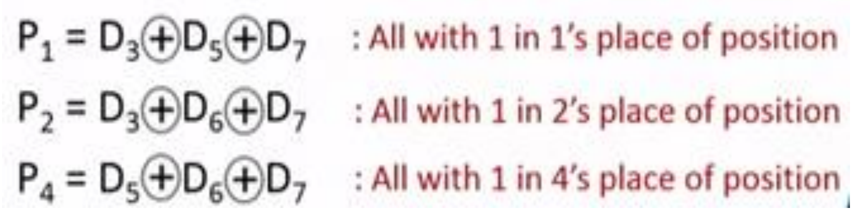
Memory Error Detection and Correction Techniques

Error detection methods used for Soft Errors

Hamming Code for Error Detection and Correction

Hamming Code Generation: Example (7,4) Code: Rate =4/7

P1	P2	D3(m3)	P4	D5(m2)	D6(m1)	D7(m0)
0		0		0	0	0
1		0		0	0	1
0		0		0	1	0
1		0		0	1	1
1		0		1	0	0
0		0		1	0	1
1		0		1	1	0
0		0		1	1	1
1		1		0	0	0
0		1		0	0	1

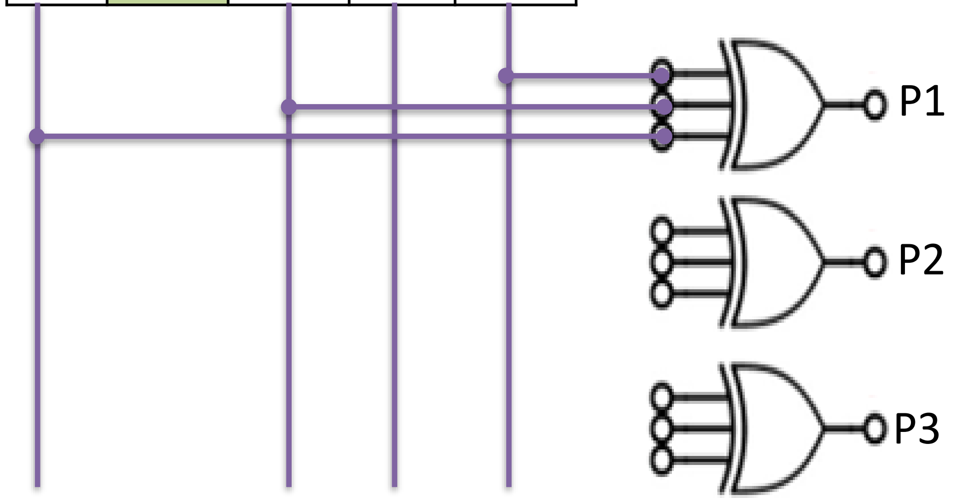


$P_1 = D_3 \oplus D_5 \oplus D_7$: All with 1 in 1's place of position

$P_2 = D_3 \oplus D_6 \oplus D_7$: All with 1 in 2's place of position

$P_4 = D_5 \oplus D_6 \oplus D_7$: All with 1 in 4's place of position

D3		D5	D6	D7
0		1	1	0



Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

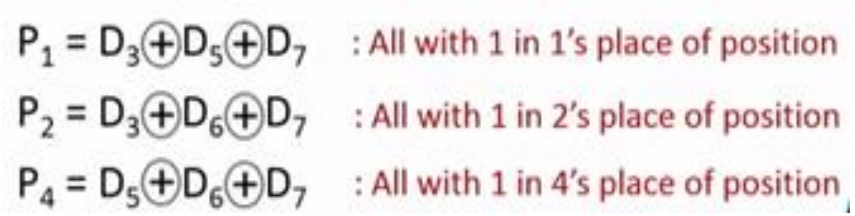
Memory Error Detection and Correction Techniques

Error detection methods used for Soft Errors

Hamming Code for Error Detection and Correction

Hamming Code Generation: Example (7,4) Code: Rate =4/7

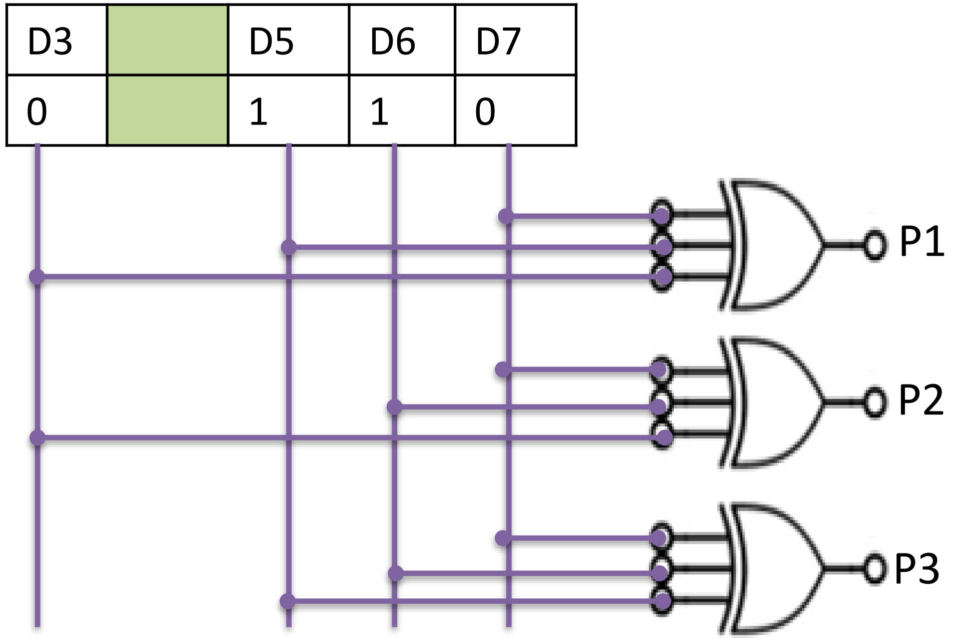
P1	P2	D3(m3)	P4	D5(m2)	D6(m1)	D7(m0)
0	0	0	0	0	0	0
1	1	0	1	0	0	1
0	1	0	1	0	1	0
1	0	0	0	0	1	1
1	0	0	1	1	0	0
0	1	0	0	1	0	1
1	1	0	0	1	1	0
0	0	0	1	1	1	1
1	1	1	0	0	0	0
0	0	1	1	0	0	1



$P_1 = D_3 \oplus D_5 \oplus D_7$: All with 1 in 1's place of position

$P_2 = D_3 \oplus D_6 \oplus D_7$: All with 1 in 2's place of position

$P_4 = D_5 \oplus D_6 \oplus D_7$: All with 1 in 4's place of position



D3		D5	D6	D7
0		1	1	0

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

Memory Error Detection and Correction Techniques

Standard Hamming code can only detect and correct a single-bit error

Error detection methods used for Soft Errors

Hamming Code for Error Detection and Correction

Hamming Code Generation: Example (7,4) Code: Rate =4/7

P1	P2	D3(m3)	P4	D5(m2)	D6(m1)	D7(m0)
0	0	0	0	0	0	0
1	1	0	1	0	0	1
0	1	0	1	0	1	0
1	0	0	0	0	1	1
1	0	0	1	1	0	0
0	1	0	0	1	0	1
1	1	0	0	1	1	0
✓ 0	0	0	1	1	1	1
✓ 1	1	1	0	0	0	0
0	0	1	1	0	0	1

} 1

Hamming Distance

1	1	0	1	0	0	1
0	1	0	1	0	1	0
1	0	0	0	0	1	1

Ex: If dmin=3 ;

a) Errors Detected is b= dmin-1 =3-1=2 .
It means **Two error can be detected.**

b) Errors corrected is
b= (dmin-1)/2 = (3-1)/2 =1
It means **ONE Error can be corrected**

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

Memory Error Detection and Correction Techniques

Error detection methods used for Soft Errors

Hamming Code for Error Detection and Correction

Error Correction

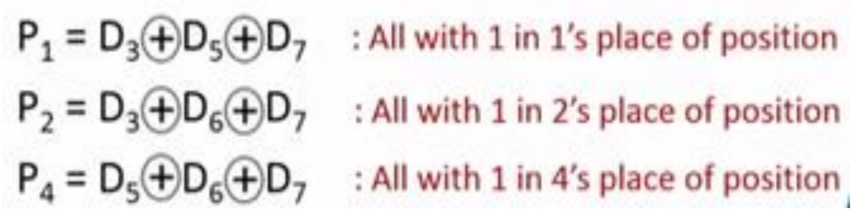
Step1 : Compute Check bits

C1 = (P1 **XOR** D3 **XOR** D5 **XOR** D7)

C2 = (P2 **XOR** D3 **XOR** D6 **XOR** D7)

C4 = (P4 **XOR** D5 **XOR** D6 **XOR** D7)

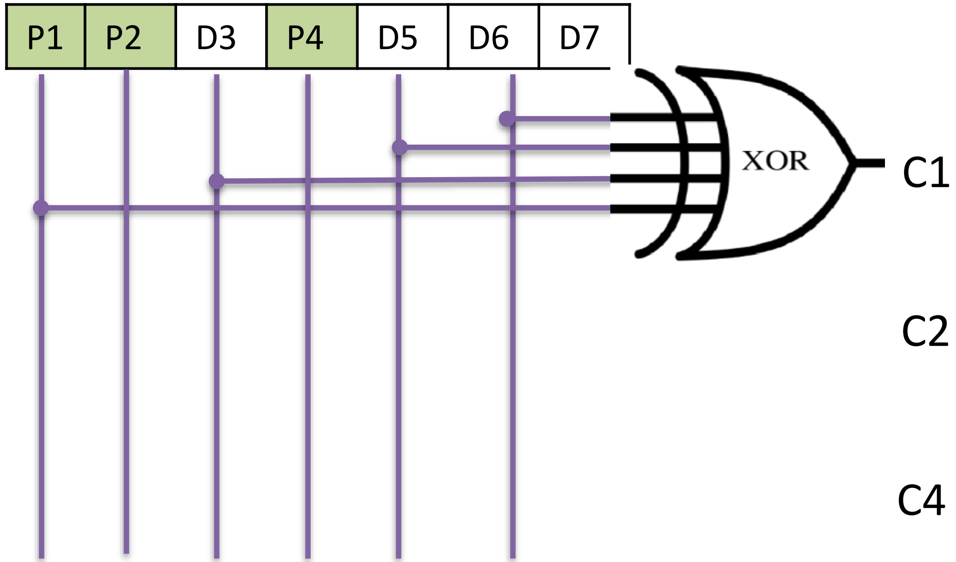
Step2 : (C4.C2.C1) = (0 0 0) ; There is NO Error else bit position of Error is given by value of (C3 C2 C1)



$P_1 = D_3 \oplus D_5 \oplus D_7$: All with 1 in 1's place of position

$P_2 = D_3 \oplus D_6 \oplus D_7$: All with 1 in 2's place of position

$P_4 = D_5 \oplus D_6 \oplus D_7$: All with 1 in 4's place of position



Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

Memory Error Detection and Correction Techniques

Error detection methods used for Soft Errors

Hamming Code for Error Detection and Correction

Error Correction

$P_1 = D_3 \oplus D_5 \oplus D_7$: All with 1 in 1's place of position

$P_2 = D_3 \oplus D_6 \oplus D_7$: All with 1 in 2's place of position

$P_4 = D_5 \oplus D_6 \oplus D_7$: All with 1 in 4's place of position

Example

1	0	0	0	0	1	1

No Error Code word (VALID)

C1 = (P1 XOR D3 XOR D5 XOR D7) = 1 xor 0 xor 0 xor 1 =0

C2 = (P2 XOR D3 XOR D6 XOR D7) = 0 xor 0 xor 1 xor 1 =0

C4 = (P4 XOR D5 XOR D6 XOR D7) = 0 xor 0 xor 1 xor 1 =0

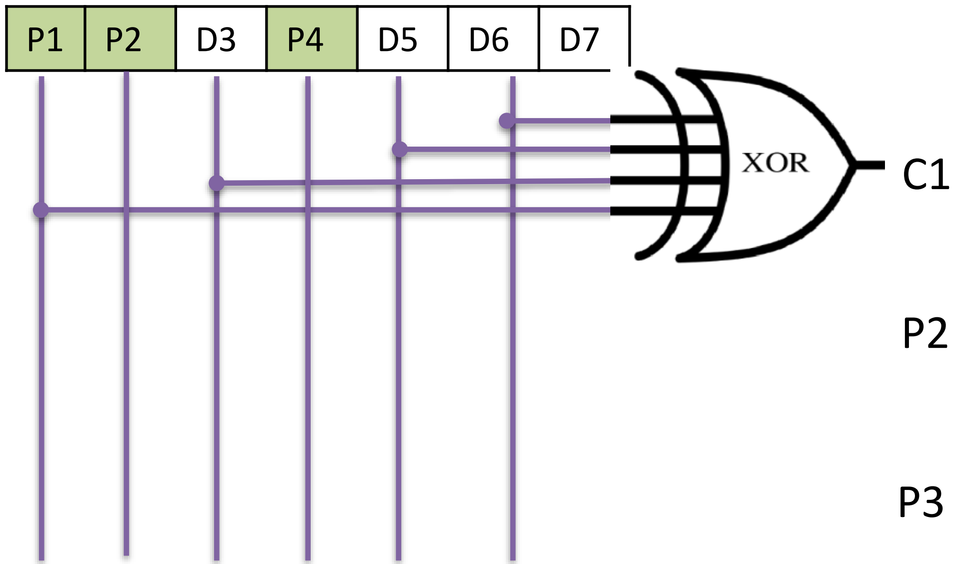
1	0	0	0	1	1	1

Erroneous Code word (INVALID)

C1 = (P1 XOR D3 XOR D5 XOR D7) = 1 xor 0 xor 1 xor 1 =1

C2 = (P2 XOR D3 XOR D6 XOR D7) = 0 xor 0 xor 1 xor 1 =0

C4 = (P4 XOR D5 XOR D6 XOR D7) = 1 xor 0 xor 1 xor 1 =1



(C3 C2 C1) = (101) means there is a error is D5 position and Need to be corrected. Simply **INVERSE the erroneous bit in the data read**

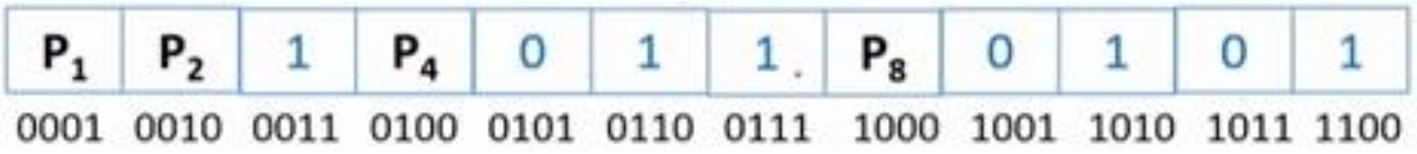
Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

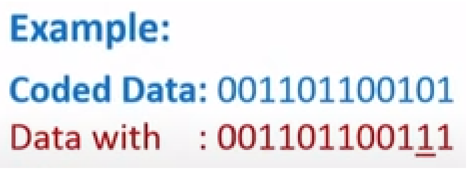
Memory Error Detection and Correction Techniques

Error detection methods used for Soft Errors

Hamming Code for Error Detection and Correction



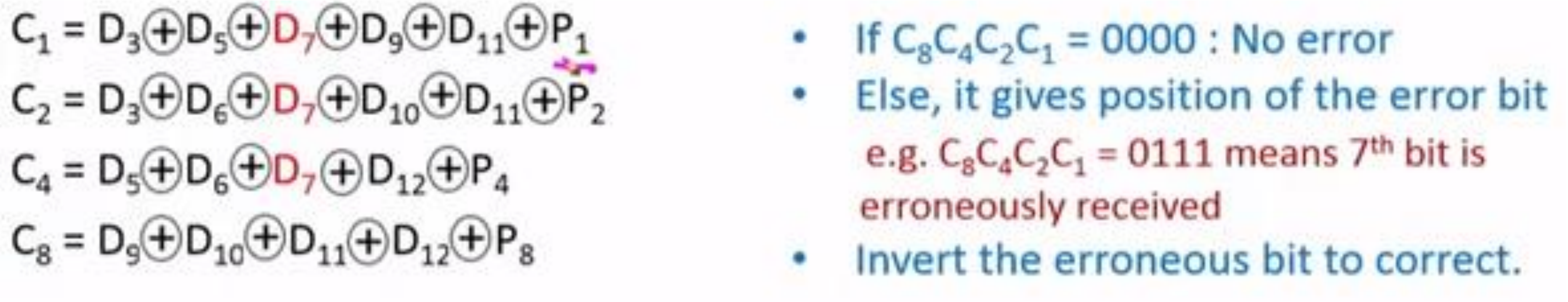
P_1	P_2	1	P_4	0	1	1	P_8	0	1	0	1
0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100



Example:

Coded Data: 001101100101

Data with : 0011011001$\underline{1}$1



$C_1 = D_3 \oplus D_5 \oplus D_7 \oplus D_9 \oplus D_{11} \oplus P_1$

$C_2 = D_3 \oplus D_6 \oplus D_7 \oplus D_{10} \oplus D_{11} \oplus P_2$

$C_4 = D_5 \oplus D_6 \oplus D_7 \oplus D_{12} \oplus P_4$

$C_8 = D_9 \oplus D_{10} \oplus D_{11} \oplus D_{12} \oplus P_8$

- If $C_8C_4C_2C_1$ = 0000 : No error
- Else, it gives position of the error bit

  e.g. $C_8C_4C_2C_1$ = 0111 means 7th bit is erroneously received
- Invert the erroneous bit to correct.

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

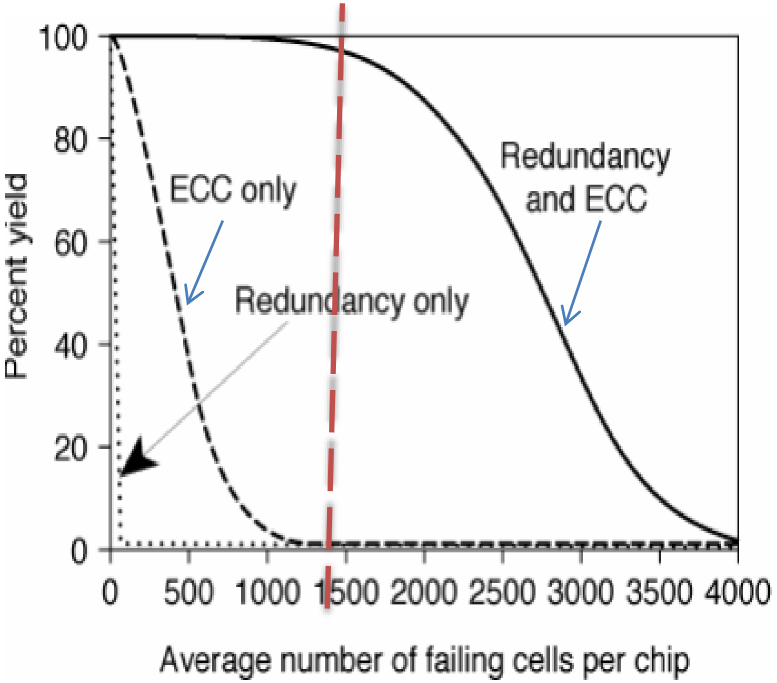
Redundancy and Error Correction

Yield

- High density and Large die size cause yield problems

$$\text{Yield} = \frac{\text{No. of Good chips on wafer}}{\text{No. of chips on wafer}} \text{x } 100\%$$

- Yield can be increased using Error Correction Code(ECC) and Redundancy



- From the graph it is clear that, the **yield can be 100% up to around 1500 failing cells per chip** **when both Redundancy and ECC techniques** are used to correct faults in memory.
- When only one of them is used, the yield decreases.

Reference :“High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

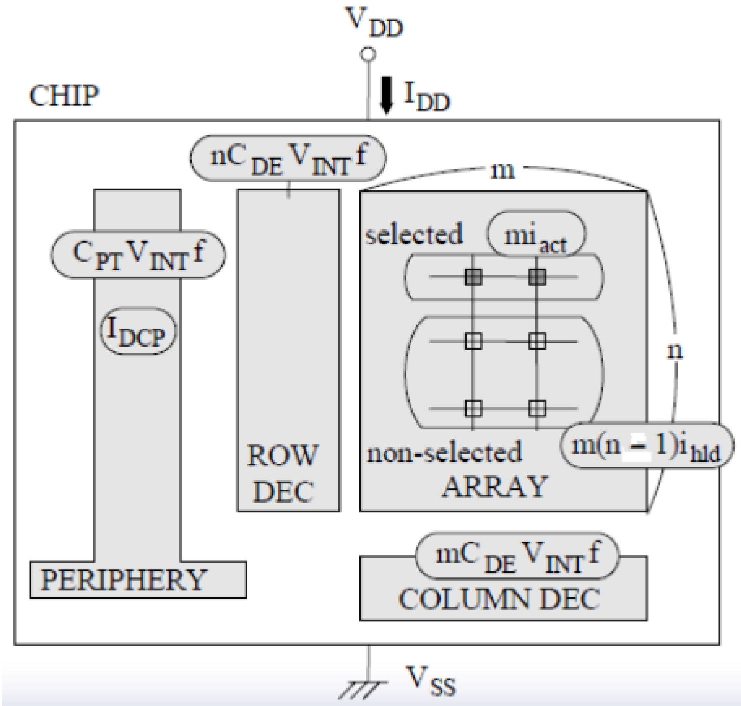
Built In Self Test (BIST)

Sources of Power Dissipation in Memories

- Power dissipation sources:
  - ✓ Memory Cell array,
  - ✓ Row and Column Decoders and
  - ✓ Peripheral Circuitry
- Memory Array with ‘m’ columns and ‘n’ rows having various components of IDD are shown

$P(t) = V_{DD} \times I_{DD}(t)$ — ①

↳ I_{DD} is function of time.



i_{act}	: Effective current of the active cell
C_{DE}	: Output Node capacitance of each Decoder
i_{hld}	: Effective data retention current of inactive cell
V_{int}	: Internal supply voltage
V_{DD}	: External supply voltage
C_{PT}	: Total capacitance of peripheral circuits
I_{DCP}	: Total static (DC) current of periphery

Power Dissipated by Row Decoder $= V_{DD}\, i_{dec}$

where $i_{dec} = n(C_{DE} \cdot V_{INT}) f = n\, Q_{DE} \times \frac{1}{t}$

$i_{dec} = (n\, i_d)$

Reference :“High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

Sources of Power Dissipation in Memories

- To reduce active power
  - ✓ Reduce capacitance
  - ✓ Reduce Internal and external voltage
  - ✓ Reduce static current
- **DRAM cell consumes more power than SRAM cell** due to **refresh operation**.
- Static cell leakage current is the major source of power dissipation in SRAM cell

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

Reliability

- The Reliability of a semiconductor device such as memory is defined as the **probability that it will perform its required function under the specified operating and environment conditions for a stated period of time**.
- A device is considered a failure if it shows **an inability to perform within its guaranteed functional and parametric** (e.g., **voltage, current, temperature**) specification limits.
- The quantitative **definition of reliability is based upon the characterization of** three important parameters:
  - ✓ Hazard rate,
  - ✓ Failure rate, and
  - ✓ Mean Time To Failure (MTTF).

Reference :“High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

Reliability

Mean Time To Failure (MTTF)

- MTTF is defined as the average time to failure for a population of devices when operated under a specific set of conditions for a given period of time.

- The failure rate λ(t) between time intervals t1 and t2 is defined as the ratio of probability that the failure occurs in this interval (given that it has not occurred prior to t1) divided by the interval length l.

- Let F(t) represent the probability of failure prior to time 't' R(t) is the probability of the device not failing prior to time 't' .

  Then R(t) = 1 – F(t)

Failure rate λ(t) : It can be expressed in terms of R(t) as

$$\lambda(t) = \frac{R(t_1) - R(t_2)}{(t_2 - t_1)R(t_1)}$$

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

Built In Self Test (BIST)

Reliability

Hazard rate h(t) : The hazard rate h(t) or instantaneous failure rate is defined as the limit of failure rate as interval length l → 0.

$$h(t) = \lim_{l \to 0} \frac{R(t_1) - R(t_2)}{lR(t_1)}$$

where l = (t2 - t1)

MTTF = 1 / λ(t2 - t1)

The failure mechanisms for memory can be broadly classified as

1) Semiconductor bulk failures
2) Dielectric failures
3) Semiconductor dielectric interface failures
4) Conductor and metallization failures
5) Assembly and packaging related failures

Reference :"High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.

THANK YOU

Mahesh Awati

Department of Electronics and Communication

mahesha@pes.edu

+91 9741172822

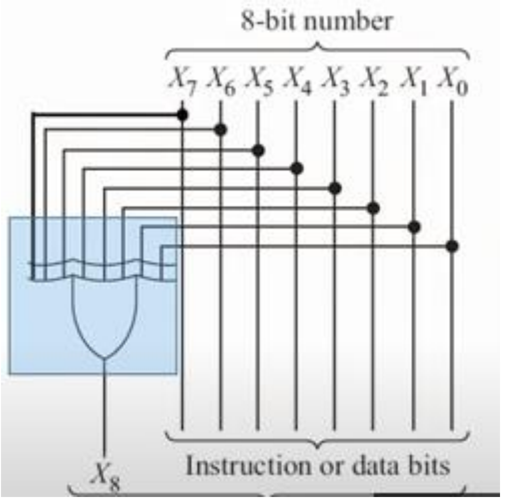
Built In Self Test (BIST)

BIST and Redundancy

HARD AND SOFT REDUNDANCY

Determine how redundancy is invoked ? Is it Soft redundancy or the hard redundancy variety

- Once the type of redundant element has been defined for a memory the next step is to determine how redundancy is invoked.
- It can be either of the soft redundancy or the hard redundancy variety.
- A hard redundancy implementation utilizes some form of fuses to store the information for memory element replacement.
- The fuses can be laser fuses, electrical fuses, or even EPROM memory .
- If a laser fuse is utilized. the correct redundancy calculation is performed and the information is uploaded off chip. This information is communicated to a laser fuser which then opens certain on-chip fuses with a laser cut. A laser fuse needs to have an opening exposed to the top passivation of the chip so that the laser



Redundant R3

Reference :“High Performance Memory Testing: Design Principles, Fault Modeling and Self Test by R.Dean Adams, Kluwer Academic Publishers, 2003.