Figure 4-1 Introduction to Memory Testing
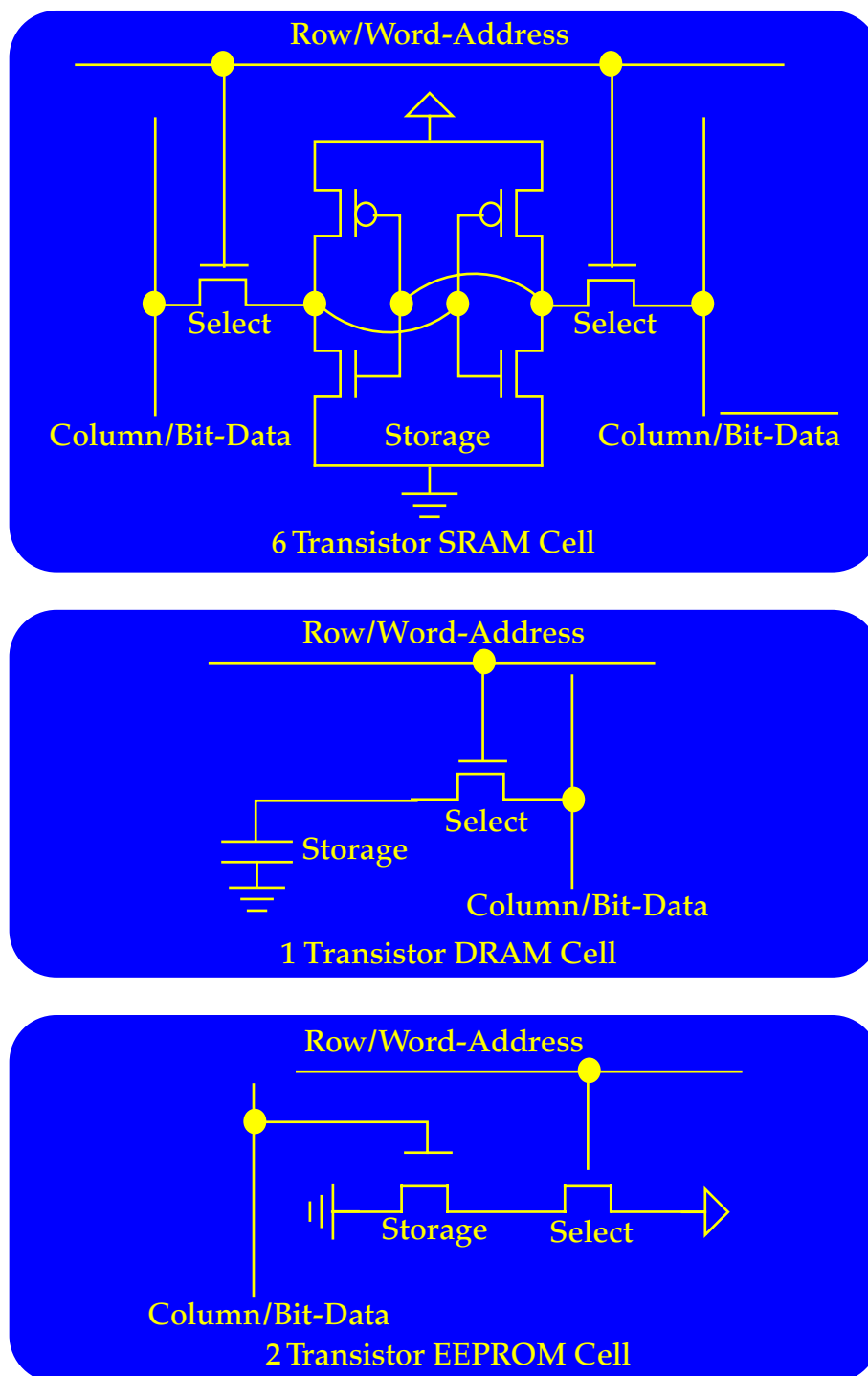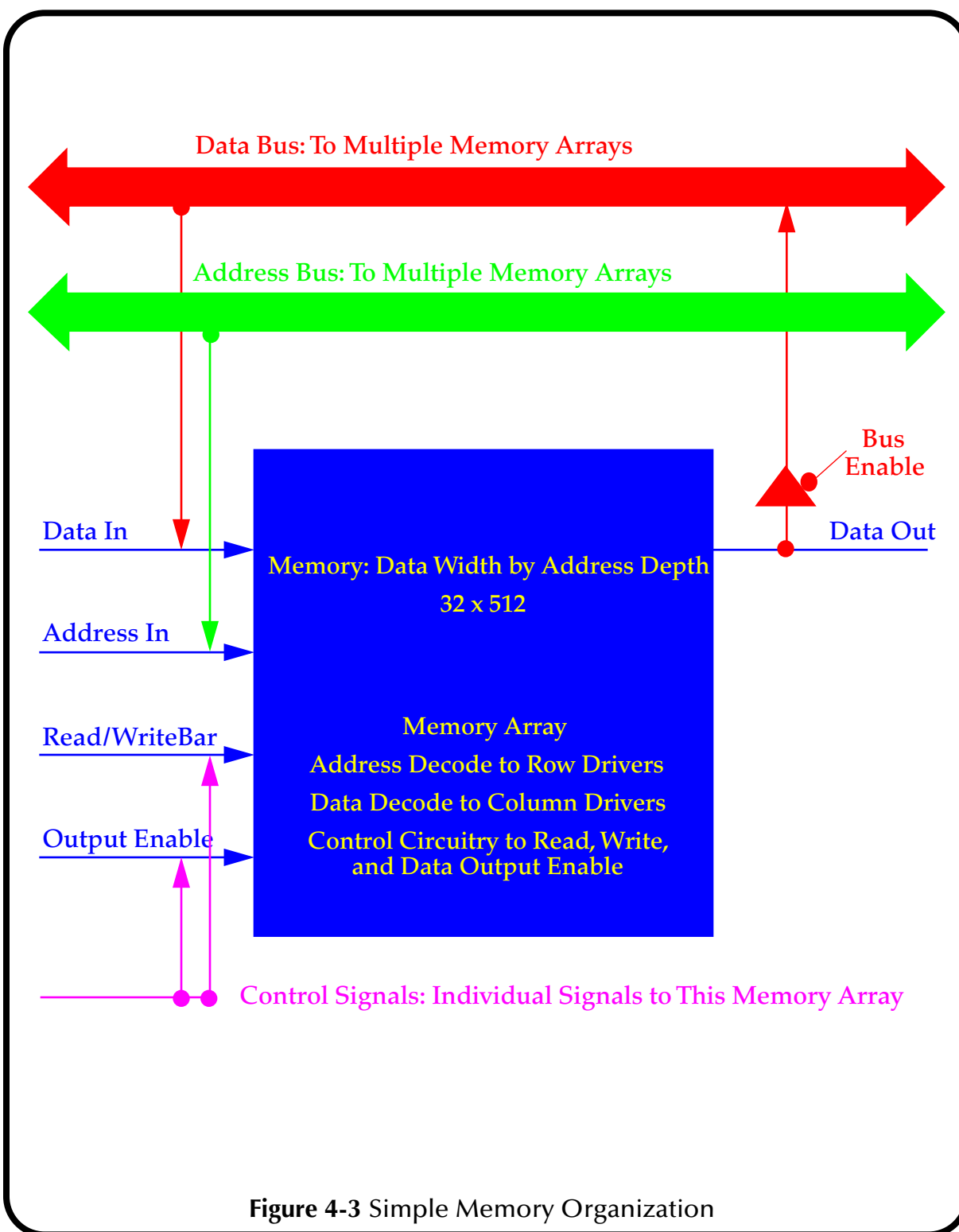
Figure 4-2 Memory Types

Data Bus: To Multiple Memory Arrays

Address Bus: To Multiple Memory Arrays

Bus Enable

Data In

Data Out

Memory: Data Width by Address Depth

32 x 512

Address In

Read/WriteBar

Memory Array

Address Decode to Row Drivers

Data Decode to Column Drivers

Output Enable

Control Circuitry to Read, Write, and Data Output Enable

Control Signals: Individual Signals to This Memory Array

Figure 4-3 Simple Memory Organization

Chip FloorPlan

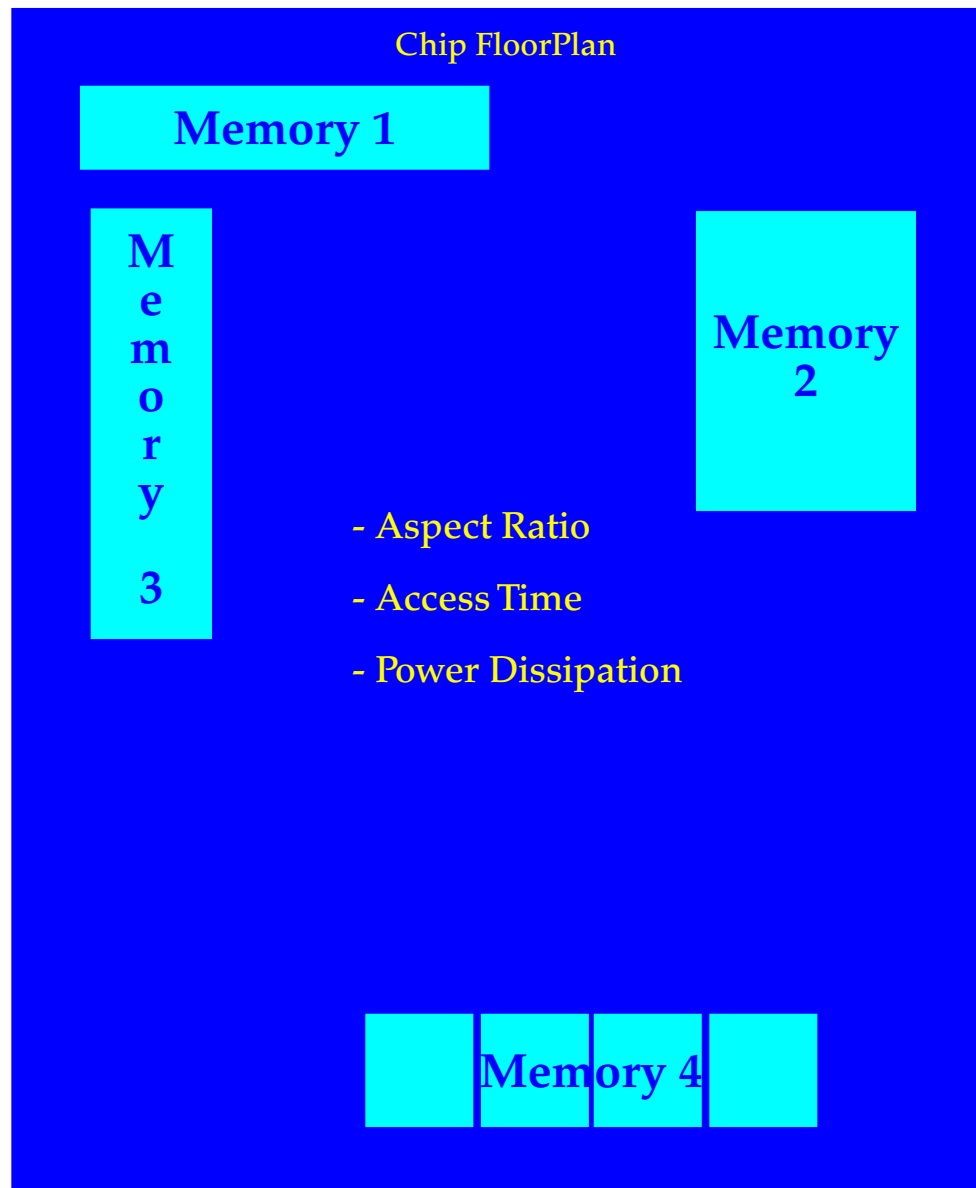Memory 1

Memory 3

Memory 2

- Aspect Ratio

- Access Time

- Power Dissipation

Memory 4

Figure 4-4 Memory Design Concerns

Chip FloorPlan

Memory 1

Memory 3

Memory 2

- Routing

- Placement & Distribution

- Overall Power Dissipation

Processor

Local Logic

Memory 4

Figure 4-5 Memory Integration Concerns

Figure 4-6 Embedded Memory Test Methods

column # —>   0   1   2   3

row # —> 0   1   0   0   1

row # —> 1   1   0   1   1

data bit cell

row # —> 2   0   1   0   1

Figure 4-7 Simple Memory Model

Data in Bit Cells

May Be Stuck-At

Logic 1 or Logic 0

word stuck-at
data value 1110

single bit stuck-at 1

address A031—>

1	0	1	1

address A032—>

1	1	1	0

address A033—>

1	0	1	0

single bit stuck-at 0

Figure 4-8 Bit-Cell and Array Stuck-At Faults

Data in Bit Cells

May Be Bridged

to Other Bit Cells

horizontal (row)
bit bridging

vertical (column)
bit bridging

random
bit bridging

word bridging
unidirectional
one-way short

word bridging
bidirectional
two-way short

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

Figure 4-9 Array Bridging Faults

Figure 4-10 Decode Faults

Data around target cell is written with complement data

Complementary
Data around
Target Cell

Address 21 = A	1	0	1	0
Address 22 = 5	0	1	0	1
Address 23 = A	1	0	1	0
Address 24 = 5	0	1	0	1

alternating 5's and A's make for a natural checkerboard pattern

Figure 4-11 Data Retention Faults

Figure 4-12 Memory Bit Mapping

	Address	0	1	2	3	4	5	6	7
Addr(00) to Addr(Max) **Write(5)-Initialize** **Increment Address**	Address 00 —>	0	1	0	1	0	1	0	1
	Address 01 —>	0	1	0	1	0	1	0	1
	Address 02 —>	0	1	0	1	0	1	0	1
Addr(00) to Addr(Max) **Read(5)-Write(A)-Read(A)** **Increment Address**	Address 03 —>	0	1	0	1	0	1	0	1
	Address 04 —>	0	1	0	1	0	1	0	1
	Address 05 —>	0	1	0	1	0	1	0	1
Addr(00) to Addr(Max) **Read(A)-Write(5)-Read(5)** **Increment Address**	Address 06 —>	0	1	0	1	0	1	0	1
	Address 07 —>	0	1	0	1	0	1	0	1
	Address 08 —>	0	1	0	1	0	1	0	1
Addr(Max) to Addr(00) **Read(5)-Write(A)-Read(A)** **Decrement Address**	Address 09 —>	0	1	0	1	0	1	0	1
	Address 10 —>	0	1	0	1	0	1	0	1
	Address 11 —>	0	1	0	1	0	1	0	1
Addr(Max) to Addr(00) **Read(A)-Write(5)-Read(5)** **Decrement Address**	Address 12 —>	0	1	0	1	0	1	0	1
	Address 13 —>	0	1	0	1	0	1	0	1
	Address 14 —>	0	1	0	1	0	1	0	1
Addr(Max) to Addr(00) **Read(5)** **Decrement Address**	Address 15 —>	0	1	0	1	0	1	0	1
	Address 16 —>	0	1	0	1	0	1	0	1
	Address 17 —>	0	1	0	1	0	1	0	1
Read (A)-------> **Write (5)** **Read (5)** **Increment Address**	Address 18 —>	1	0	1	0	1	0	1	0
	Address 19 —>	1	0	1	0	1	0	1	0
	Address 20 —>	1	0	1	0	1	0	1	0
March C+ Algorithm	Address 21 —>	1	0	1	0	1	0	1	0
	Address 22 —>	1	0	1	0	1	0	1	0
	Address 23 —>	1	1	1	0	1	0	1	0

Memory Array with 24 Addresses
with Algorithm at Read (A) Stage

Figure 4-13 Algorithmic Test Generation

**Boundary at some level
of scanned registration
or "pipelining" away
from the memory
array**

Data

**Memory

Array**

Data

**Detection of
incoming
signals**

**Control of
outgoing
signals**

Address

Control

**scan-memory
boundary**

**Minimum Requirement
Detection up to Memory Input
and Control of Memory Output**

**Concern: the Logic between
the Scan Test Area and the
Memory Test Area Is not
Adequately Covered**

**Non-Scanned Registration inside the Boundary
but Before the Memory Test Area Results in
a Non-Overlap Zone**

Figure 4-14 Scan Boundaries

**The Memory Array is modeled
for the ATPG Engine so the
ATPG Tool can use the memory
to observe the inputs
and control the outputs**

Data In

Din   Dout

Data Out

Memory

Array

Address

ATPG

Ain

Model

Control

Read/Write

Scan
Architecture

Figure 4-15 Memory Modeling

**Boundary at some level
is blocked off
as if the memory was
cut out of the circuit**

Scan Mode

**Control of
outgoing
signals**

Data In

Gated Data Out

**Detection of
incoming
signals**

Memory

Array

can be

Address

Multiplexed Data Out

removed

from

netlist for

**All Registers
are in the
scan chain
architecture**

ATPG purposes

Control

scan black-box
boundary

**Observe-only registers
used for detection of memory
input signals**

**Gate or Multiplexor is used
to Block—fix to a known
value—the Memory
Output Signals**

Figure 4-16 Black Box Boundaries

**Boundary at some level
is blocked off
as if the memory was
cut out of the circuit**

**Input is passed
to output as the
form of output
control**

Data In

**Detection of
incoming
signals**

Memory
array
can be
removed
from
netlist for
ATPG purposes

Bypass Data Out

Address

Control

scan black-box
boundary

**Observe-only registers
used for detection of memory
input signals**

**Multiplexor is used to
pass the input directly
to the output**

Figure 4-17 Memory Transparency

Detection of incoming data signals done here

Boundary at some level is blocked off as if the memory was cut out of the circuit

Input is passed to output with registration

Data In

Bypass Data Out

Memory array can be removed from netlist for ATPG purposes

In ideal sense, timing should also be matched

Address

Control

scan black-box boundary

Observe-only registers not needed on data since register emulates memory

Register and multiplexor is used to emulate memory timing and output

Figure 4-18 The Fake Word Technique

Data Bus: Possibly to Multiple Memory Arrays

Address Bus: Possibly to Multiple Memory Arrays

Data In

Data Out

Memory: data width by address depth

32 x 512

Address

Read/WriteB

Memory Array

Address Decode to Row Drivers

Data Decode to Column Drivers

Output Enable

Control Circuitry to Read, Write,
and Data Output Enable

Control Signals: Individual Signals to This Memory Array

Test Must Access the Data, Address, and Control Signals
in order to Test This Memory

Figure 4-19 Memory Test Needs

Chip Level

Algorithm Controller

Address Generator

Data Generator

Memory Array(s)

Comparator

Invoke

Retention

Debug

Done

Fail

Debug_data

INPUTS
Invoke: Start BIST
Retention: Pause BIST and Memory Clocking
Debug: Enable BIST Bitmap Output

OUTPUTS
Fail: A Memory Has Failed a BIST Test
Done: Operation of BIST Is Complete
Debug_data: Debug Data Output

OPERATIONS
Address: Ability to Apply Address Sequences
Data: Ability to Apply Different Data Sequences
Algorithm: Ability to Apply Algorithmic Control Sequences
Comparator: Ability to Verify Memory Data

Figure 4-20 Memory BIST Requirements

INPUTS
Invoke: invoke the BIST (apply muxes and release reset)
Retention: enable retention algorithm and pause
Release: discontinue and release pause
Bitmap: enable bitmap output on fail occurrence

OUTPUTS
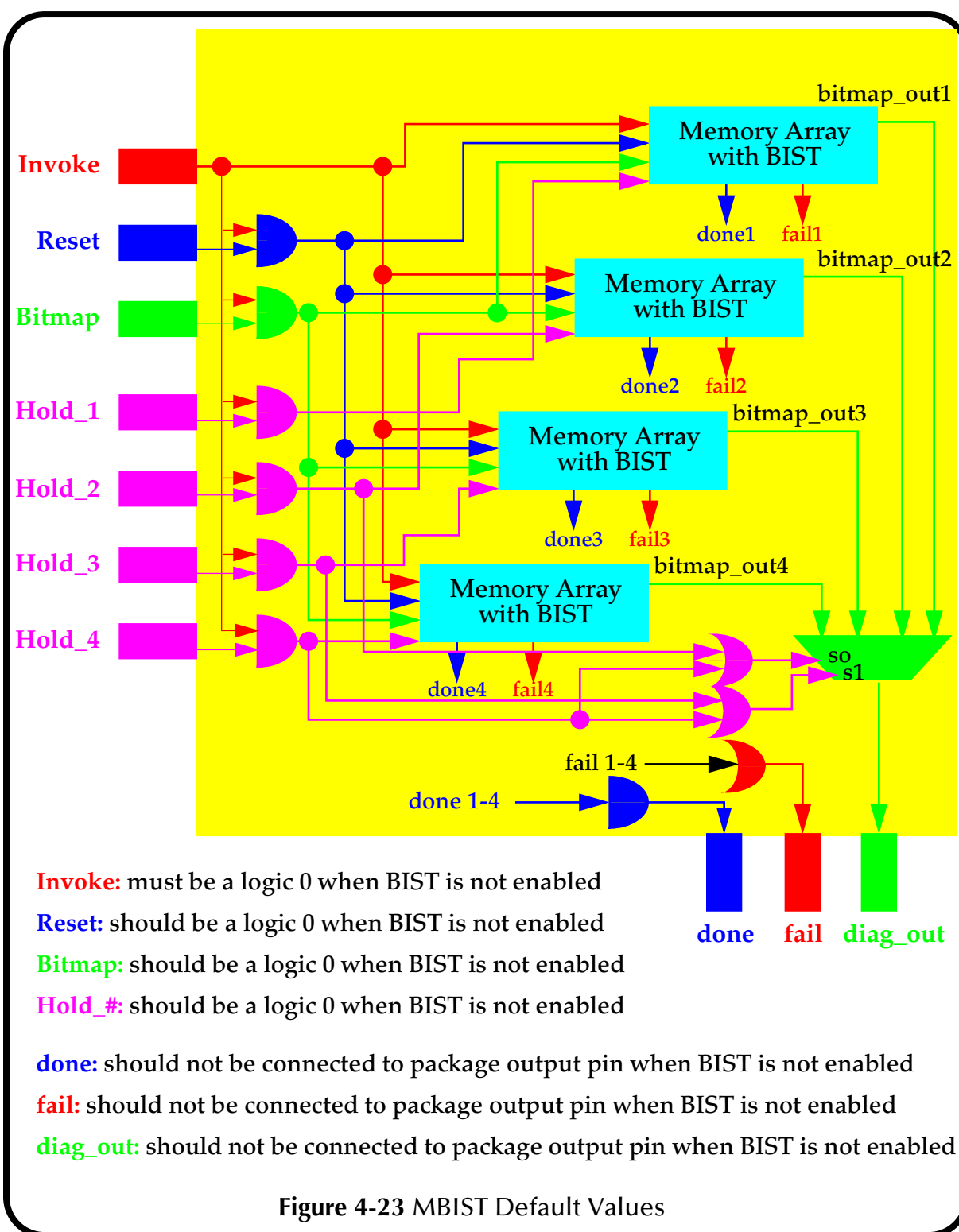**Fail:** sticky fail flag—dynamic under bitmap
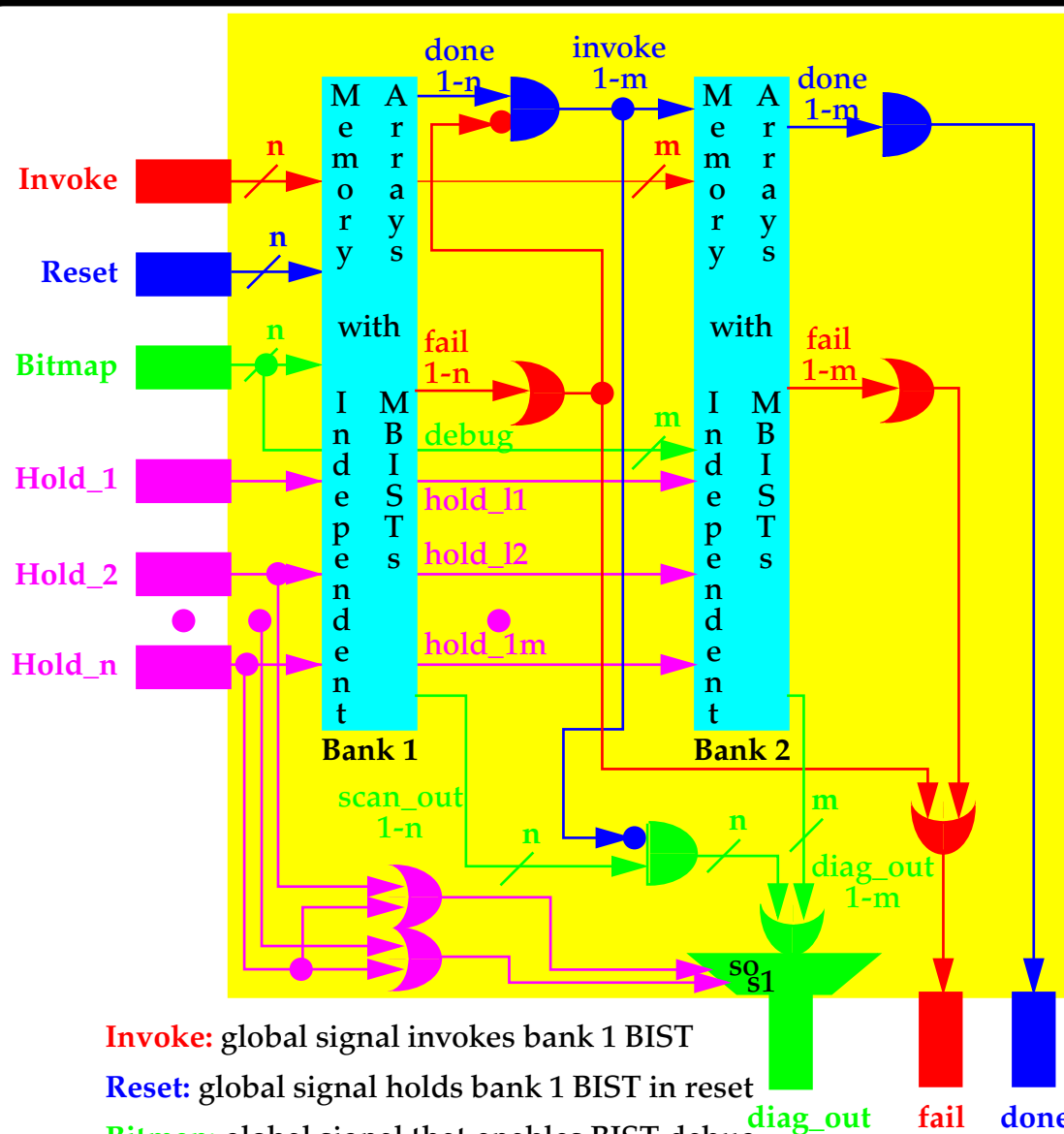**Done:** operation of BIST is complete
Bitmap_out: fail data under bitmap
Hold_out: indication of pause

Figure 4-21 An Example Memory BIST

Figure 4-22 MBIST Integration Issues

Invoke: a global signal to invoke all BIST units

Reset: a global signal to hold all BIST units in reset

Bitmap: a global signal to put all BIST units in debug mode

Hold_#: individual hold signals to place memories in retention
        or to select which memory is displayed during debug

done: all memory BISTs have completed

fail: any memory BIST has detected a fault or a failure

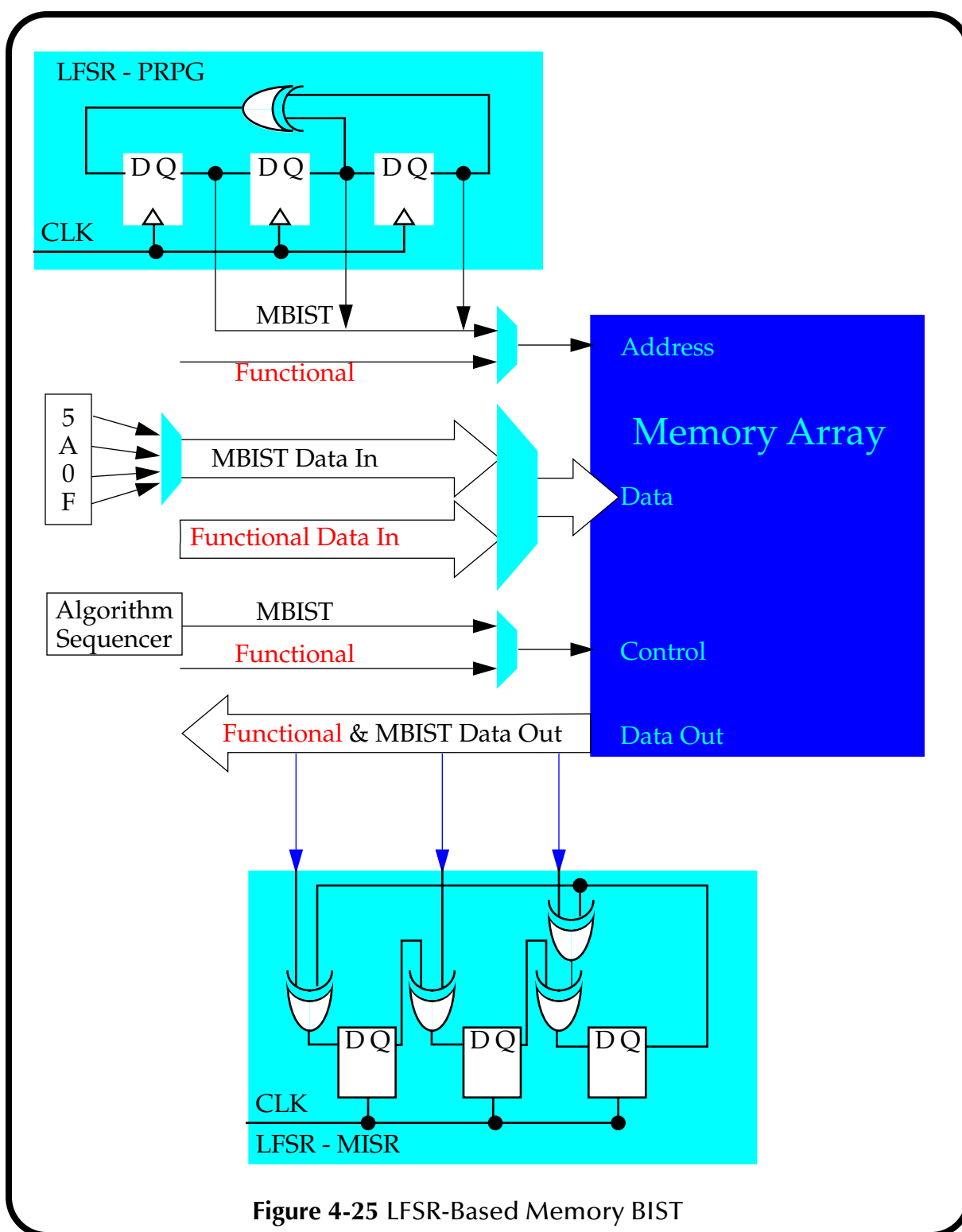**diag_out:** the memory BIST not in hold mode will present debug data

Invoke: must be a logic 0 when BIST is not enabled
Reset: should be a logic 0 when BIST is not enabled
Bitmap: should be a logic 0 when BIST is not enabled
Hold_#: should be a logic 0 when BIST is not enabled

done: should not be connected to package output pin when BIST is not enabled
fail: should not be connected to package output pin when BIST is not enabled
diag_out: should not be connected to package output pin when BIST is not enabled

Figure 4-23 MBIST Default Values

Invoke: global signal invokes bank 1 BIST

Reset: global signal holds bank 1 BIST in reset

Bitmap: global signal that enables BIST debug

Hold_#: paired hold signals to place memories in retention
           or to select which memory is displayed during debug

done: bank n memory BISTs have completed

fail: any memory BIST has detected a fault or a failure

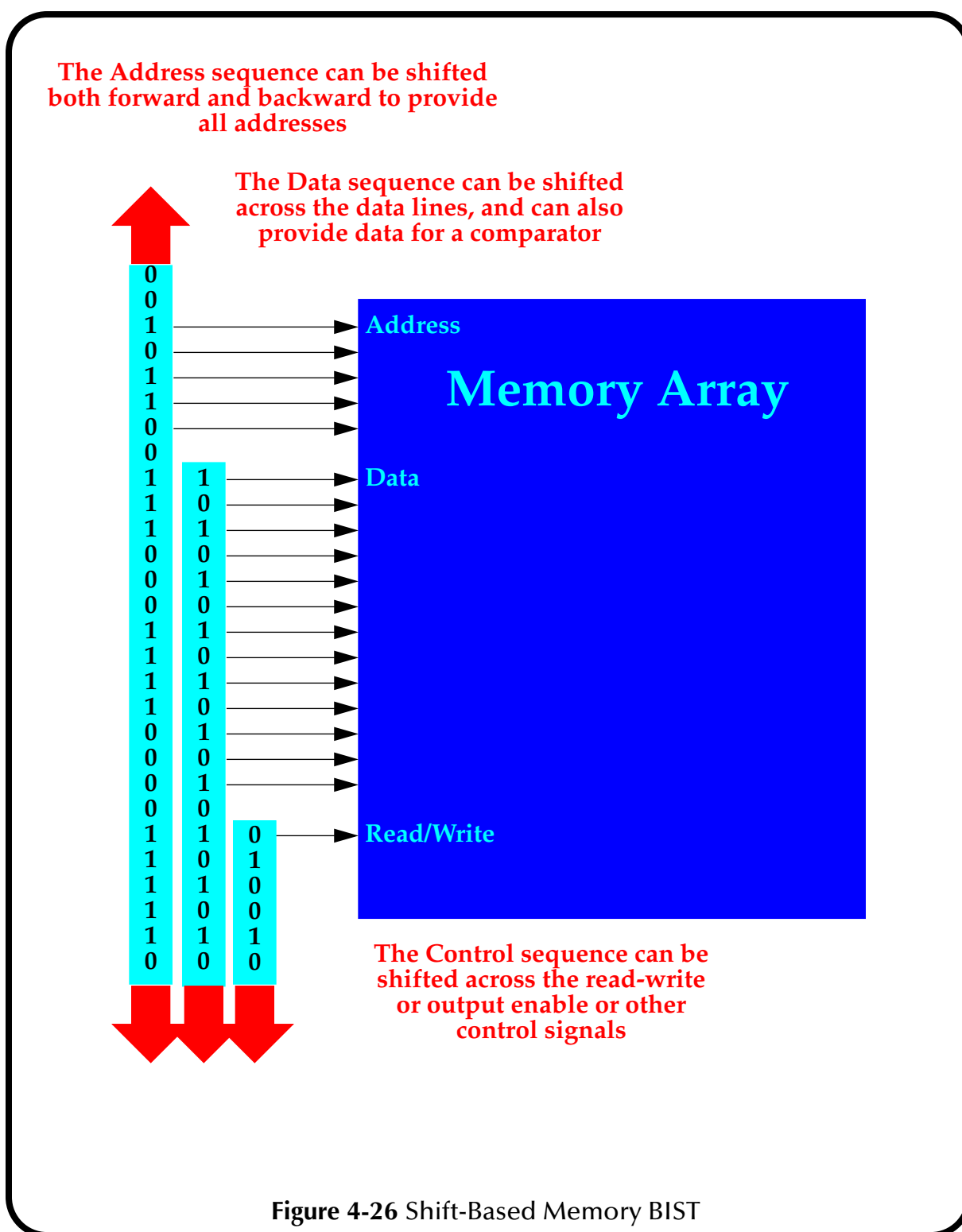**diag_out:** the memory BIST not in hold will present debug data
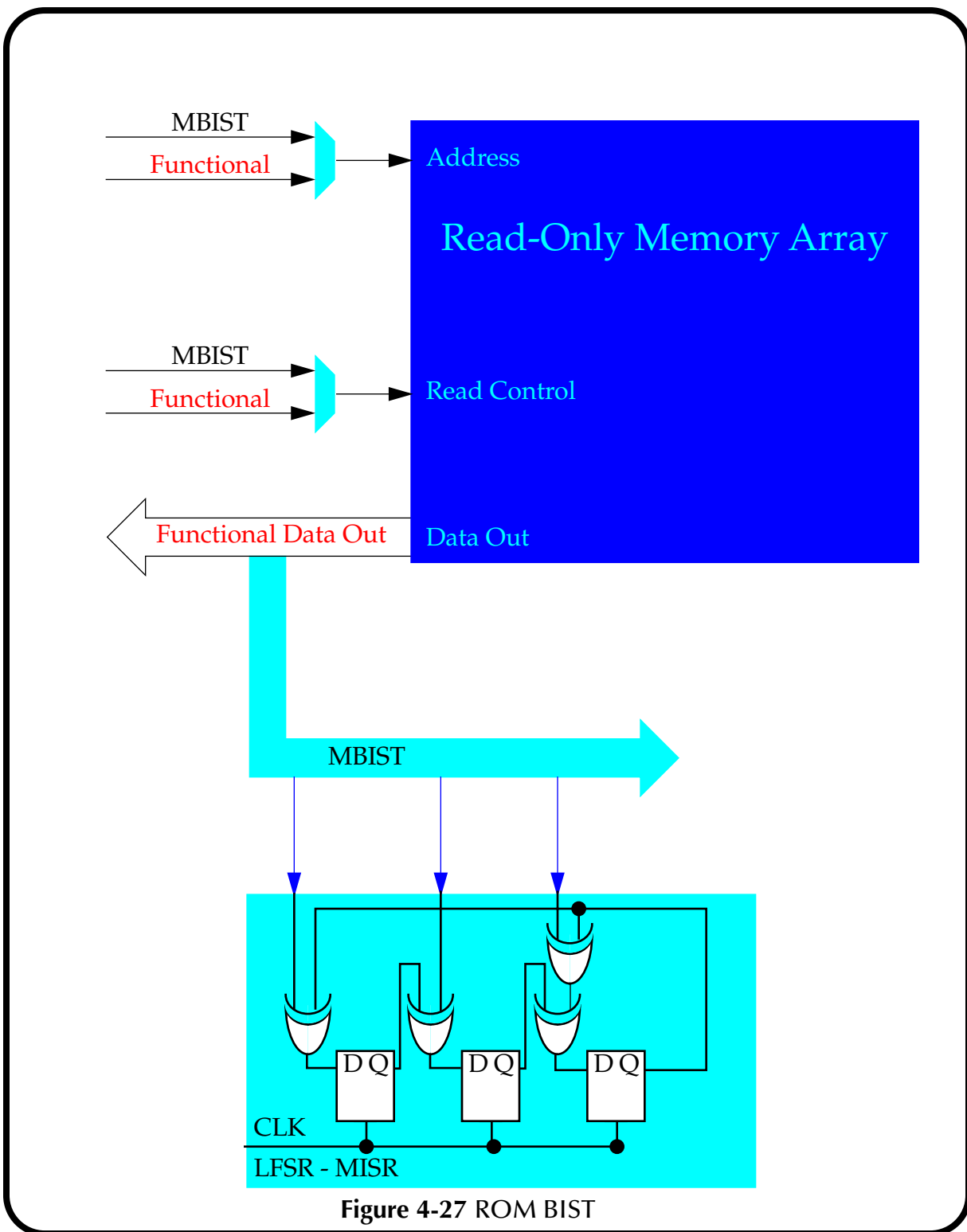
Figure 4-24 Banked Operation

Figure 4-25 LFSR-Based Memory BIST

The Address sequence can be shifted both forward and backward to provide all addresses

The Data sequence can be shifted across the data lines, and can also provide data for a comparator

Address

Memory Array

Data

Read/Write

The Control sequence can be shifted across the read-write or output enable or other control signals

Figure 4-26 Shift-Based Memory BIST

Figure 4-27 ROM BIST

Memory Testing Fundamentals Summary

Memory Testing Is Defect-Based

Memory Testing Is Algorithmic

Different Types of Memories—Different Algorithms

A Memory Fault Model Is Wrong Data on Read

Memory Testing Relies on Multiple-Clue Analysis

A Memory Test Architecture May CoExist with Scan

A Memory Can Block Scan Test Goals

Modern Embedded Memory Test Is BIST-Based

BIST Is the Moving of the Tester into the Chip

BIST-Based Testing Allows Parallelism

Parallel Testing Impacts Retention Testing

Parallel Testing Impacts Power Requirements

Parallel Testing Requires Chip-Level Integration

Figure 4-28 Memory Test Summary