Testability of VLSI

Lecture 09:
Testing of Memory

By Dr. Sanjay Vidhyadharan

sanjayvidhyadharan.in

Types of Memory

1.Dynamic Random Access Memory **(DRAM)** has the highest possible density but a slow access time of 20 *ns*. Bits are stored as charge on a single capacitor, but the memory must be refreshed, typically every 2, 4, or 6 *ms*, if information is not to be lost. 32 GB RAMS available.

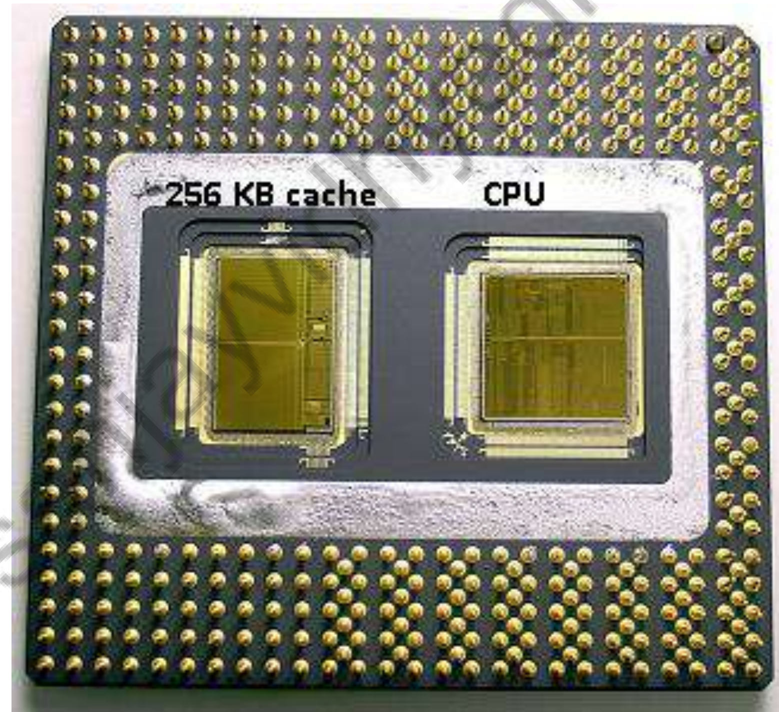
Synchronous DRAM (SDRAM)
DDR (Double Data Rate) DRAM Pipeling hence faster DDR does not wait for completion of previous read/write operation to continue other operation.

sanjayvidhyadharan.in

Types of Memory

***2. Static Random Access Memory* (SRAM)** has the fastest possible speed, with a 2 *ns* access time. Bits are stored in cross-coupled latches, and the memory need not be refreshed.

***3. Cache DRAM* (CDRAM)** combines both SRAM and DRAM on the same chip, in order to accelerate block transfer between the SRAM cache and the slow DRAM.

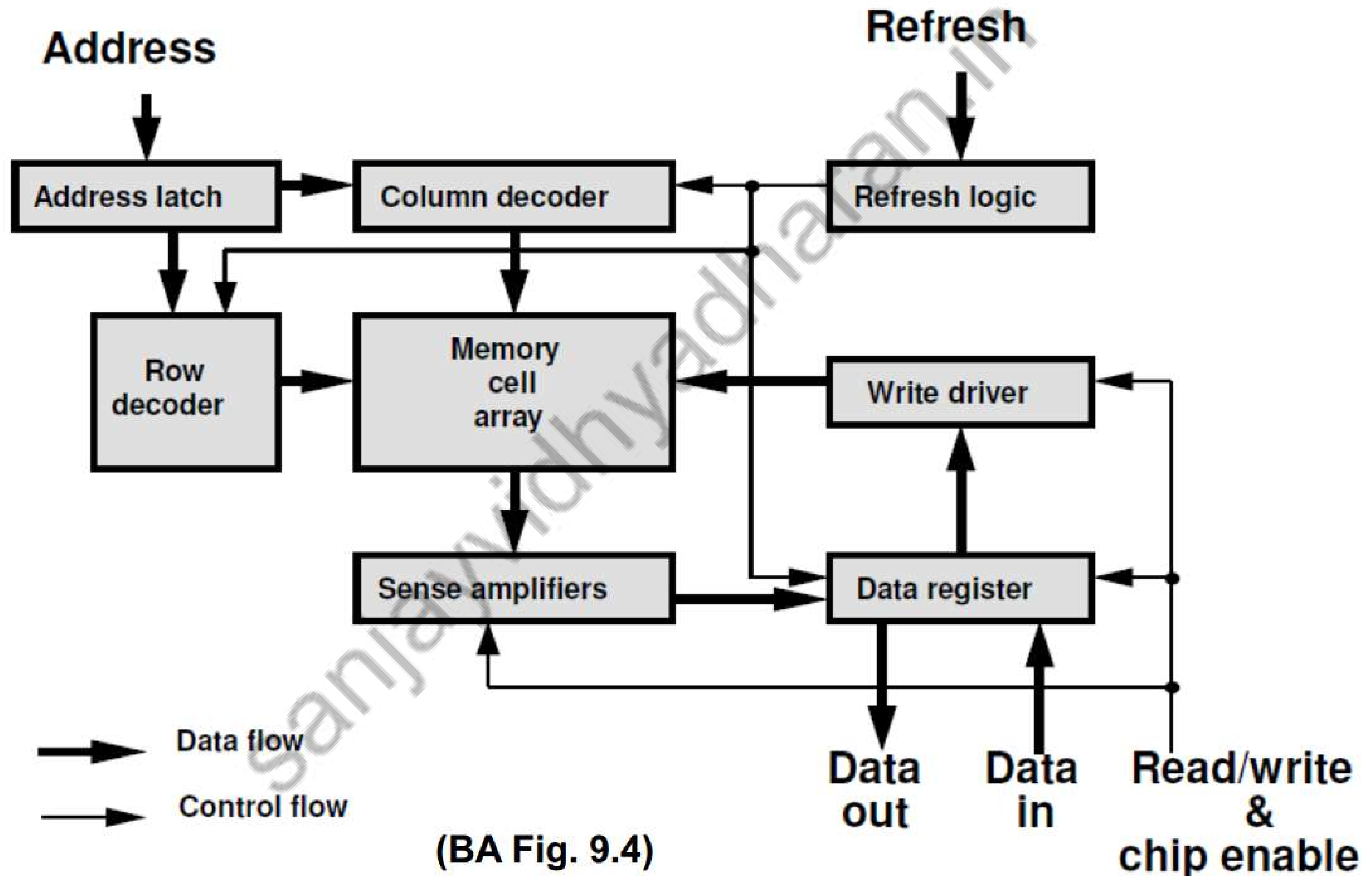


Types of Memory

4. Read-Only Memories **(ROMs/EPROMs/EEPROMs)** have every bit content programmed by the presence or absence of a transistor at manufacturing time, and do not lose information when power is shut off.

sanjayvidyadharan.in

Memory Organization



(BA Fig. 9.4)

sanjayvidhyadharan.in

Memory Testing

- Whole Chip not discarded for a single fault. Fault detecting and correction
- Static faults
  - Cells, Decoder etc
- Dynamic Fault
  - Write time, Access Time and Data retention time

sanjayvidhyadharan.in

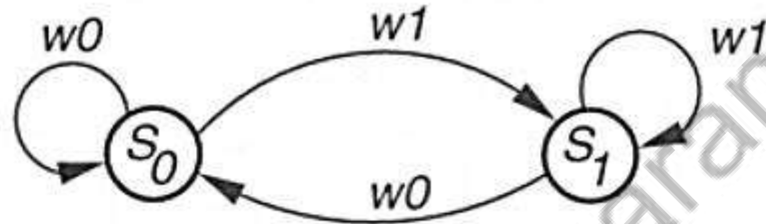
Fault Models

- Stuck-at faults
- Transition faults
- Coupling faults
- Neighborhood pattern sensitive fault
- Address decoder fault

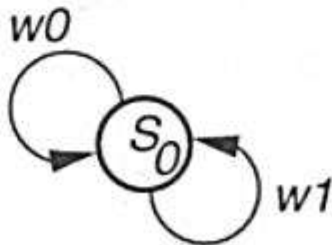
sanjayvidhyadharan.in

Fault Models

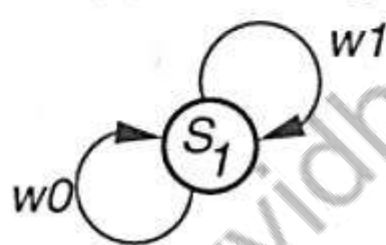
1. Stuck-at faults



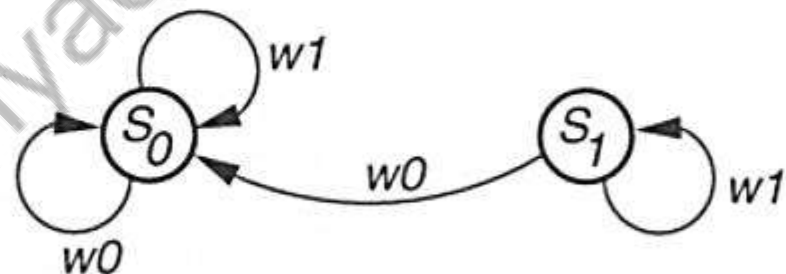
(a) State diagram of a good cell.



(b) SA0 fault.



(c) SA1 fault.



(d) <↑/0> transition fault.

2. Transition Faults. The *transition fault* (TF) is a special case of the SAF, in which a cell fails to make a $0 \to 1$ (up) transition or a $1 \to 0$ (down) transition when it is written. An *up transition fault* is denoted as $<\uparrow /0>$, while a *down transition fault* is denoted as $<\downarrow /1>$.

sanjayvidhyadharan.in

March Test Notation

Symbol	Meaning
r	Memory action: A read operation
w	Memory action: A write operation
r0	Memory action: Read a 0 from the memory location
r1	Memory action: Read a 1 from the memory location
w0	Memory action: Write a 0 to the memory location
w1	Memory action: Write a 1 to the memory location
$\uparrow$	Write a 1 to a cell containing 0 or the cell has a rising transition
$\downarrow$	Write a 0 to a cell containing 1 or the cell has a falling transition
$\updownarrow$	Complement the cell contents
$\Uparrow$	Increasing memory addressing order
$\Downarrow$	Decreasing memory addressing order
$\Updownarrow$	Addressing order can be either increasing or decreasing
$\rightarrow$	Write a 0 to a cell containing a 0
$^{\rightarrow}$	Write a 1 to a cell containing a 1
$\Rightarrow$	Write value x to a cell already containing x
$\forall$	Denotes any memory write operation: $\forall \in \{\uparrow, \downarrow, \updownarrow, \rightarrow, ^{\rightarrow}, \Rightarrow\}$

sanjayvidhyadharan.in

March Test Notation

MATS+ $\{ \Updownarrow (w0); \Uparrow (r0, w1); \Downarrow (r1, w0) \}$.

```
M0:  { March element ⇕ (w0) }
     for cell := 0 to n − 1 (or any other order) do
     begin
       write 0 to A [ cell ];
     end;

M1:  { March element ⇑ (r0, w1) }
     for cell := 0 to n − 1 do
     begin
       read A [ cell ]; { Expected value = 0 }
       write 1 to A [ cell ];
     end;

M2:  { March element ⇓ (r1, w0) }
     for cell := n − 1 down to 0 do
     begin
       read A [ cell ]; { Expected value = 1 }
       write 0 to A [ cell ];
     end;
```

M. S. Abadir and J. K. Reghbati, “Functional Testing of Semiconductor Random Access Memories,” *ACM Computing Surveys*, vol. 15, no. 3, pp. 175–198, Sept. 1983.

sanjayvidhyadharan.in

Fault Models

3. Coupling Faults

A *coupling fault* (CF) means that a transition in memory bit *j* causes an unwanted change in memory bit *i*. The *2-coupling fault* is a coupling fault involving two cells. A write operation that generates an↑ or↓ in transition in cell *j* changes the contents of cell *i*. The 2-coupling fault is a special case of the *k-coupling fault*, which has the 2-coupling fault behavior with respect to cells *i* and *j*, except that faulty behavior occurs only when another $k - 2$ cells are in a particular state.

3.1 Inversion Coupling Faults:

↑ or ↓ transition in cell *j* inverts the contents of cell *i*. Cell *i* is said to be *coupled* to cell *j*, which is the *coupling cell*. We use the notation $<\uparrow;\updownarrow>$ for C_i and C_j, where the ↕ means that cell C_i contents were inverted. The two possible CFin types are $<\uparrow;\updownarrow>$ and $<\downarrow;\updownarrow>$.

sanjayvidhyadharan.in

Fault Models

3.1 Inversion Coupling Faults:

$\uparrow$ or $\downarrow$ transition in cell j inverts the contents of cell i. Cell i is said to be *coupled* to cell j, which is the *coupling cell.* We use the notation $<\uparrow;\updownarrow>$ for C_i and C_j, where the $\updownarrow$ means that cell C_i contents were inverted. The two possible CFin types are $<\uparrow;\updownarrow>$ and $<\downarrow;\updownarrow>$.

Not all linked CFins can be detected by march tests

Consider three cells *i, j,* and *k* (with address relationships Address(*i*) < Address(*j*) < Address(*k*).) Cell *k* is coupled to cell *i* and to cell *j*, and both *i* and *j* are visited either before or after *k* is visited by a march element.

Using the sequence and/or its reverse. In this case: (i) The two CFins will mask each other for any march element marching ‘up’, and (ii) Neither will be triggered for an element marching ‘down’. Therefore, the march test fails to detect the linked CFins.

For all cells that are coupled, each should be read after a series of possible CFins may have occurred (due to writing into the coupling cells), and the number of coupled cell transitions must be odd (to prevent the CFins from masking each other.)

sanjayvidhyadharan.in

Fault Models

3.2 Idempotent Coupling Faults:

Idempotent Coupling Faults. An *idempotent coupling fault* (CFid) is where an ↑ or ↓ transition in cell C_j sets cell C_i to 0 or 1. This is denoted as $<\uparrow;0>$ or $<\uparrow;1>$, depending on whether cell i is set to 0 or 1, for a rising transition for cell j. The other two idempotent coupling faults are $<\downarrow;0>$ and $<\downarrow;1>$. A test to *detect* all CFids has this necessary condition:

3.3 Dynamic Coupling Faults:

A *dynamic coupling fault* (CFdyn) occurs between cells in different words. A read or write operation on one cell forces the contents of the second cell either to 0 or 1. This is a more general case of the Cfid.

CFdyn can be sensitized by any read or write operation, where as a CFid can only be sensitized by a writing a change (transition write operation) to the coupling cell. We denote a CFdyn as $< r0|w0;0 >$ where | denotes the *or* of the read and write operations, which must be done to the coupling cell [688]. There are four CFdyn faults: $< r0|w0;0 >$, $< r0|w0;1 >$, $< r1|w1;0 >$, and $< r1|w1;1 >$.

sanjayvidhyadharan.in

Fault Models

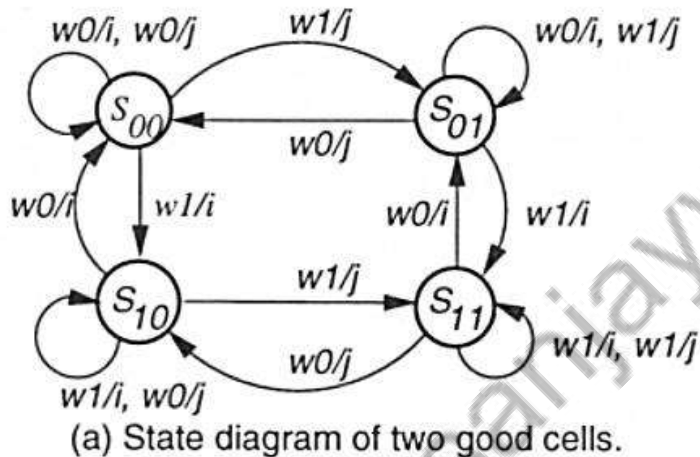
4. Bridging Faults

Bridging Faults. A *bridging fault* (BF) is a short circuit between two or more cells or lines. It is a bidirectional fault, so either cell/line can affect the other cell/line. A 0 or 1 state of the coupling cell causes the fault, rather than a coupling cell transition. With the *AND bridging fault* (ABF), the logical bridge value is the AND of the shorted cells/lines. The four possible ABFs are < 0,0/0,0 >, < 0,1/0,0 >, < 1,0/0,0 >, and < 1,1/1,1 >. The notation is the good machine values for cells i and j, followed (after the slash) by their bad machine values. With the *OR bridging fault* (OBF), the logical bridge value is the OR of the shorted cells/lines. The four possible OBFs are < 0,0/0,0 >, < 0,1/1,1 >, < 1,0/1,1 >, and < 1,1/1,1 >.

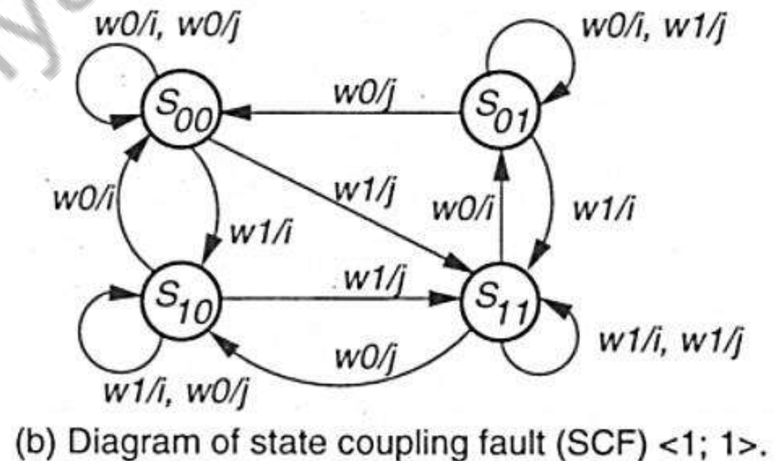
sanjayvidhyadharan.in

Fault Models

5. State Coupling Faults. The *state coupling fault* (SCF) [194] is where the coupling cell/line j is in a given state y that forces the coupled cell/line i into state x. The four SCFs are $< 0; 0 >$, $< 0; 1 >$, $< 1; 0 >$, and $<1;1>$. Figure 9.9 [442] shows a Mealy machine model of the state coupling fault, along with a more complete model of the transition fault [106, 107, 169].



(a) State diagram of two good cells.



(b) Diagram of state coupling fault (SCF) <1; 1>.

sanjayvidhyadharan.in

Fault Models

6. Neighborhood Pattern Sensitive Coupling Faults. In a *pattern sensitive fault* (PSF), the content of cell i (or the ability of cell i to change) is influenced by the contents of all other memory cells, which may be either a pattern of 0s and 1s or a pattern of transitions. The PSF is the most general *k-coupling fault*, where $k = n$ (all of the memory.) The *neighborhood* is the total number of cells involved in this fault, where the *base cell* is the cell-under-test, and the *deleted neighborhood*

Figure 9.10: Type-1 neighborhood definition.

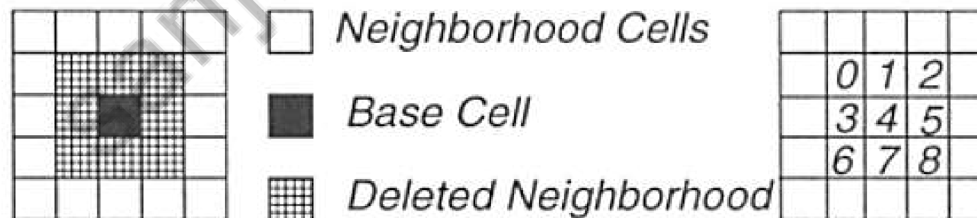


Figure 9.11: Type-2 neighborhood definition.

ELECTRICAL ELECTRONICS COMMUNICATION INSTRUMENTATION

Fault Models

6. **Neighborhood Pattern Sensitive Coupling Faults.** In a *pattern sensitive fault* (PSF), the content of cell i (or the ability of cell i to change) is influenced by the contents of all other memory cells, which may be either a pattern of 0s and 1s or a pattern of transitions. The PSF is the most general *k-coupling fault*, where $k = n$ (all of the memory.) The *neighborhood* is the total number of cells involved in this fault, where the *base cell* is the cell-under-test, and the *deleted neighborhood*

6.1 *Active NPSF* (ANPSF) [645] (also called *dynamic)*, the base cell changes due to a change in the pattern of the deleted neighborhood. One deleted neighborhood cell has a transition, while the rest of the neighborhood (including the base cell) has a given pattern.

6.2 *Passive NPSF* (PNPSF) means that a certain neighborhood pattern prevents the base cell from changing

sanjayvidhyadharan.in

Fault Models

7. Address Decoder Faults

An *address decoder fault* (AF) represents an address decoding error, in which we assume that the decoder logic does not become sequential [491, 659]. We also assume that the fault is the same during both read and write operations. We discuss only bit-oriented memory, in which each word contains only.

Van de Goor [688] classifies these faults into four cases:

Fault 1: No cell is accessed for a certain address,
Fault 2: No address can access a certain cell,
Fault 3: With a particular address, multiple cells are simultaneously accessed, and
Fault 4: A particular cell can be accessed with multiple addresses.

A march test satisfying Conditions 1 and 2 in Table 9.8 detects all address decoder faults

Table 9.8: Conditions for address decoder fault detection.

Condition	March element	Condition	March element
1	$\Uparrow$ (rx, . . . ,w$\overline{x}$)	2	$\Downarrow$ (r$\overline{x}$, . . . ,wx)

sanjayvidhyadharan.in

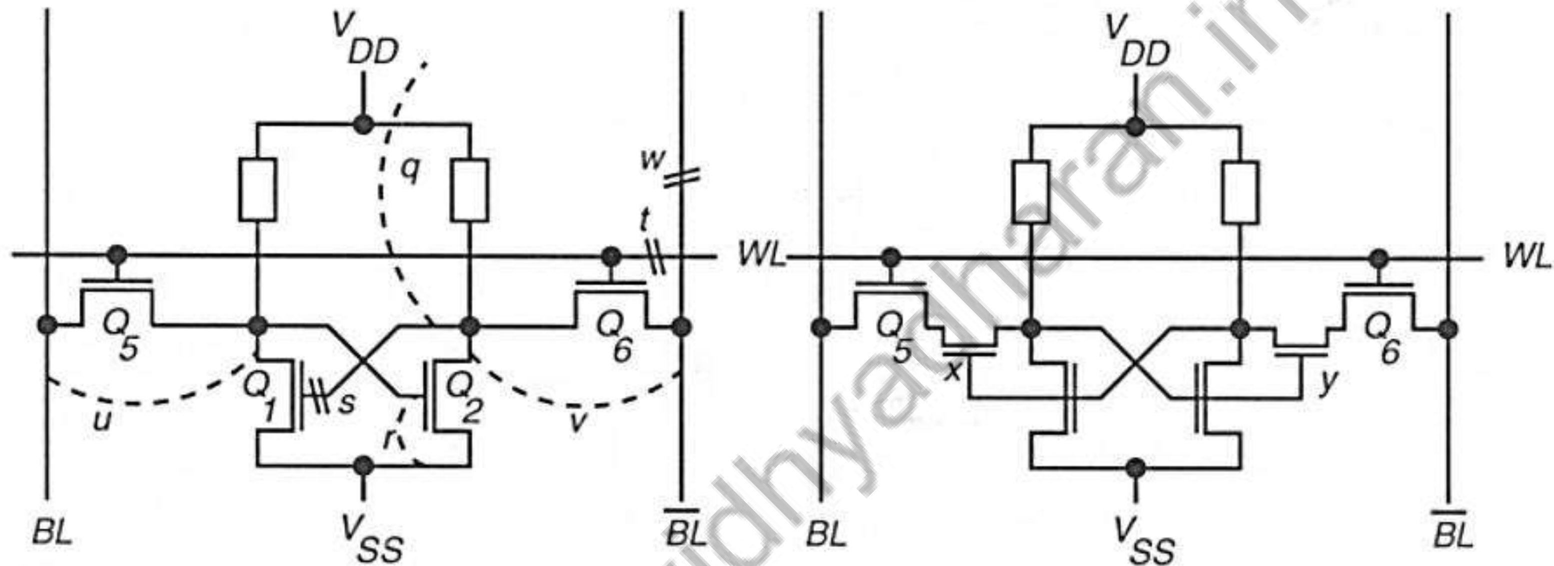
Reduced Functional Fault Modeling

Table 9.9: Mapping of functional faults onto reduced functional faults.

Reduced functional fault		Functional fault
SAF	a	Cell stuck
SAF	b	Driver stuck
SAF	c	Read/write line stuck
SAF	d	Chip-select line stuck
SAF	e	Data line stuck
SAF	f	Open circuit in data line
CF	g	Short circuit between data lines
CF	h	Crosstalk between data lines
AF	i	Address line stuck
AF	j	Open circuit in address line
AF	k	Shorts between address lines
AF	l	Open circuit in decoder
AF	m	Wrong address access
AF	n	Multiple simultaneous address access
TF	o	Cell can be set to 0 but not to 1 (or vice versa)
NPSF	p	Pattern sensitive cell interaction

sanjaywidhyadharan.in

Reduced Functional Fault Modeling



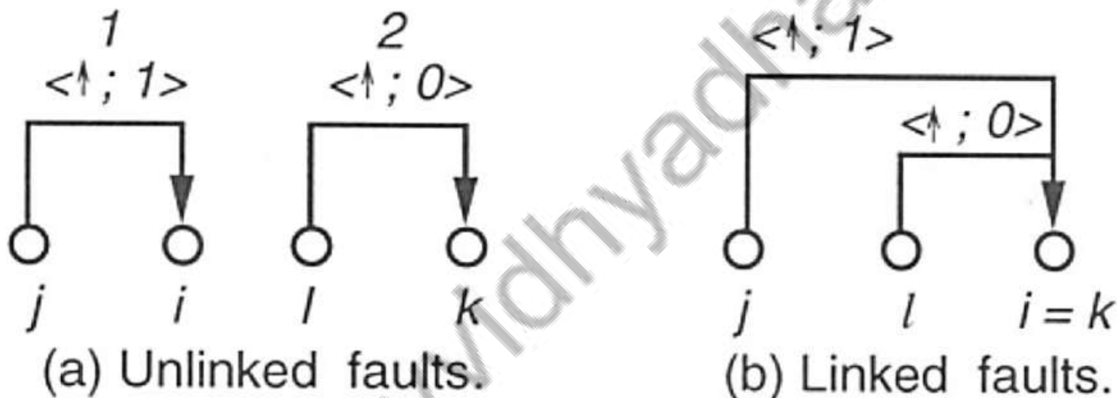
Defect u (a short between the true node and BL) will pull BL down if the cell contains a 0, but will not affect BL if the cell contains 1. Cells along BL will enter state 0 when the cell with defect u contains 0. This is the state coupling fault $< 0;0 >$. Defect v (short between inverse node and $\overline{BL}$) is similar, as is the state coupling fault $< 1;1 >$. Defect w (open $\overline{BL}$) prevents cells after the open defect from passing a logic 0 value on $\overline{BL}$. Cells after this defect containing 0 will be correctly read. When they contain 1, the result depends on the type of read circuit (if it is single-input, this is a SAF.)

sanjayvidhyadharan.in

Multiple Fault Models

Linkage. Faults may also be *linked* meaning that a fault may influence the behavior of other faults. *Unlinked* faults do not influence the behavior of other faults

Fault Masking Example



(a) Unlinked faults. (b) Linked faults.

Goor's [688] march test

$$\{ M0 :\Updownarrow (w0); M1 :\Uparrow (r0, w1); M2 :\Downarrow (w0, w1); M3 :\Uparrow (r1, w0, w1) \}.$$

sanjayvidhyadharan.in

Functional RAM Testing with March Tests

Algorithm	Fault coverage								Operation count
	SAF	AF	TF	CF in	CF id	CF dyn	SCF	Linked faults	
MATS	All	Some							$4 \cdot n$
MATS+	All	All							$5 \cdot n$
MATS++	All	All	All						$6 \cdot n$
MARCH X	All	All	All	All					$6 \cdot n$
MARCH C−	All	All	All	All	All	All	All		$10 \cdot n$
MARCH A	All	All	All	All				All linked CFids, Some CFins linked with CFids	$15 \cdot n$
MARCH Y	All	All	All	All				All TFs linked with CFins	$8 \cdot n$
MARCH B	All	All	All	All				All linked CFids, All TFs linked with CFids or CFins, Some CFins linked with CFids	$17 \cdot n$

sanjayvidhyadharan.in

Functional RAM Testing with March Tests

Algorithm	Description	Ref.
MATS	$\{ \Updownarrow (w0); \Updownarrow (r0, w1); \Updownarrow (r1) \}$	[362, 490]
MATS+	$\{ \Updownarrow (w0); \Uparrow (r0, w1); \Downarrow (r1, w0) \}$	[4, 736]
MATS++	$\{ \Updownarrow (w0); \Uparrow (r0, w1); \Downarrow (r1, w0, r0) \}$	[688]
MARCH X	$\{ \Updownarrow (w0); \Uparrow (r0, w1); \Downarrow (r1, w0); \Updownarrow (r0) \}$	[688]
MARCH C−	$\{ \Updownarrow (w0); \Uparrow (r0, w1); \Uparrow (r1, w0);$ $\Downarrow (r0, w1); \Downarrow (r1, w0); \Updownarrow (r0) \}$	[429]
MARCH A	$\{ \Updownarrow (w0); \Uparrow (r0, w1, w0, w1); \Uparrow (r1, w0, w1);$ $\Downarrow (r1, w0, w1, w0); \Downarrow (r0, w1, w0) \}$	[646]
MARCH Y	$\{ \Updownarrow (w0); \Uparrow (r0, w1, r1); \Downarrow (r1, w0, r0); \Updownarrow (r0) \}$	[688]
MARCH B	$\{ \Updownarrow (w0); \Uparrow (r0, w1, r1, w0, r0, w1);$ $\Uparrow (r1, w0, w1); \Downarrow (r1, w0, w1, w0); \Downarrow (r0, w1, w0) \}$	[646]

sanjayvidhyadharan.in

Functional RAM Testing with March Tests

MATS+ detection of cell (2, 1) SA0 fault.

$$\{ \Updownarrow (w0); \Uparrow (r0, w1); \Downarrow (r1, w0) \}$$

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

(a) Good machine after M0.

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

(b) Good machine after M1.

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

(c) Good machine after M2.

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

(d) Bad machine after M0.

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

(e) Bad machine after M1.

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

(f) Bad machine after M2.

sanjayvidhyadharan.in

Functional RAM Testing with March Tests

MATS+ detection of cell (2, 1) SA1 fault.

$\{ \Updownarrow (w0); \Uparrow (r0, w1); \Downarrow (r1, w0) \}$

0	0	0
0	0	0
0	0	0

(a) Good machine after M0.

1	1	1
1	1	1
1	1	1

(b) Good machine after M1.

0	0	0
0	0	0
0	0	0

(c) Good machine after M2.

0	0	0
1	0	0
0	0	0

(d) Bad machine after M0.

1	1	1
1	1	1
1	1	1

(e) Bad machine after M1.

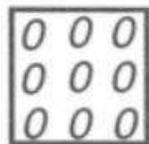
0	0	0
1	0	0
0	0	0

(f) Bad machine after M2.

sanjayvidhyadharan.in

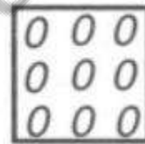
Functional RAM Testing with March Tests

MATS+ detection of cell (2, 1) multiple address decoder faults.
where cell (2, 1) is unaddressable, and address (2,1) maps instead to an access of cell (3, 1).

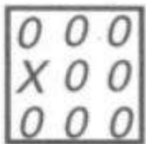


(a) Good machine after M0.

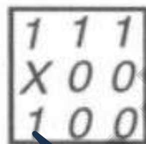
(b) Good machine after M1.



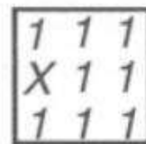
(c) Good machine after M2.



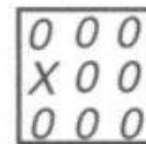
(d) Bad machine after M0.



(e) Bad machine after M1 for cell (2, 1).



(f) Bad machine after M1.



(g) Bad machine after M2.

$$\{ \Updownarrow (w0); \Uparrow (r0, w1); \Downarrow (r1, w0) \}$$

Gets detected as r0 is done before wo

sanjayvidhyadharan.in

Neighborhood Pattern-Sensitive Faults

Assumptions. We always assume that read operations of memory cells are fault-free in the NPSF testing algorithms

1.There are two different possible values for the base cell (0 and 1),

2. $k - 1$ ways of choosing the deleted neighborhood cell which must undergo one of two possible transitions

3. 2^{k-2} possibilities for the remaining neighborhood cell

4. The total number of *active neighborhood patterns* $2 \times (k-1) \times 2 \times 2^{k-2} = (k-1) \times 2^k$.

2 states of base cell

One cell selected at time for transition

Two transitions of the selected neighbouring cell

states of cell not under transition

For each of the deleted neighborhood patterns, the two possible transitions and must be verified .

$$k \times 2^k.$$

sanjayvidhyadharan.in

Neighborhood Pattern-Sensitive Faults

Table 9.16: *Active* (ANP) and *passive* (PNP) neighborhood patterns.

b	00000000000000001111111111111111	00000000000000001111111111111111
0	↑↑↑↑↑↑↑↑↓↓↓↓↓↓↓↓↑↑↑↑↑↑↑↑↓↓↓↓↓↓↓↓	00001111000011110000111100001111
1	00001111000011110000111100001111	00110011001100110011001100110011
3	00110011001100110011001100110011	01010101010101010101010101010101
4	01010101010101010101010101010101	↑↑↑↑↑↑↑↑↓↓↓↓↓↓↓↓↑↑↑↑↑↑↑↑↓↓↓↓↓↓↓↓
b	00000000000000001111111111111111	↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↑↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓
0	00001111000011110000111100001111	00000000111111110000000011111111
1	↑↑↑↑↑↑↑↑↓↓↓↓↓↓↓↓↑↑↑↑↑↑↑↑↓↓↓↓↓↓↓↓	00001111000011110000111100001111
3	00110011001100110011001100110011	00110011001100110011001100110011
4	01010101010101010101010101010101	01010101010101010101010101010101
b	00000000000000001111111111111111	
0	00001111000011110000111100001111	
1	00110011001100110011001100110011	
3	↑↑↑↑↑↑↑↑↓↓↓↓↓↓↓↓↑↑↑↑↑↑↑↑↓↓↓↓↓↓↓↓	
4	01010101010101010101010101010101	

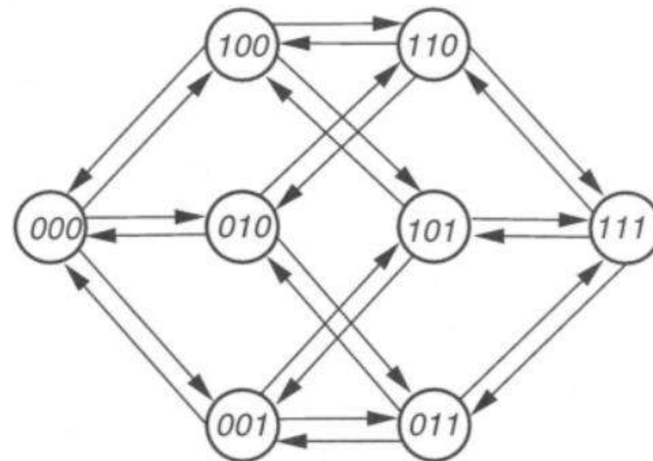
Neighborhood Pattern-Sensitive Faults

Optimal Write Sequences. It is essential to minimize the number of *writes* during NPSF testing, in order to obtain the shortest possible test

A *Hamiltonian sequence* is used for writing during *static neighborhood pattern sensitive fault* (SNPSF) Patterns in a *k*-bit Hamiltonian sequence differ by only 1 bit from their preceding pattern, as this minimizes the number of writes needed to generate the patterns. The *Gray code* is a Hamiltonian sequence.

PNPSFs and ANPSFs are tested with an *Eulerian sequence.*

There is an arc between two nodes, if and only if they differ by exactly one bit. An *Eulerian sequence* traverses each arc in the graph exactly once, while a *Hamiltonian sequence* traverses each node in the graph exactly once.

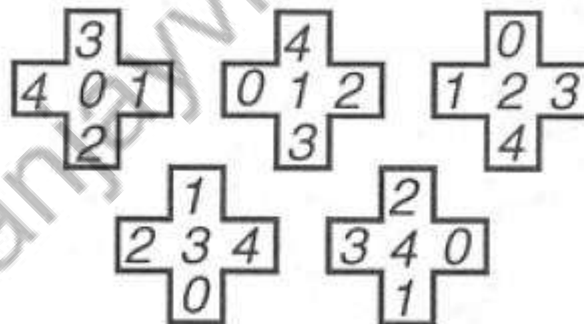


sanjayvidhyadharan.in

Neighborhood Pattern-Sensitive Faults

Testing Neighborhoods Simultaneously. When a cell is written, we change *k* different neighborhoods (Type-1 or Type-2.) We wish to test the neighborhoods simultaneously, using the *tiling* and *two-group* methods

Tiling Method. The *tiling method* totally covers memory with non-overlapping neighborhoods. Reduces the pattern length from $n \times 2^k$ patterns to $\frac{n}{k} \times 2^k$ patterns.

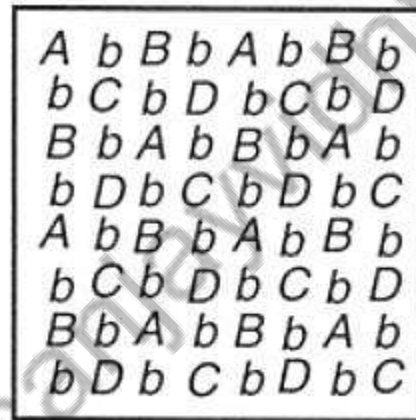


sanjayvidhyadharan.in

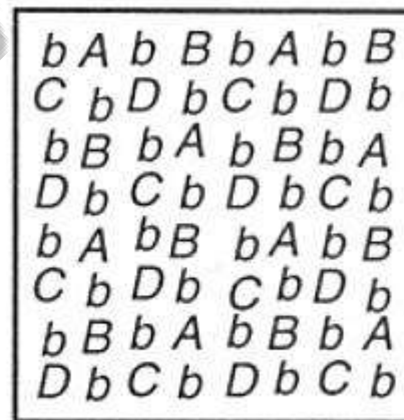
Neighborhood Pattern-Sensitive Faults

Testing Neighborhoods Simultaneously. When a cell is written, we change *k* different neighborhoods (Type-1 or Type-2.) We wish to test the neighborhoods simultaneously, using the *tiling* and *two-group* methods

Two-Group Method. For the *two-group method*, a cell is simultaneously a base cell in one group and a deleted cell in the other group, and vice versa



(a) Labels of cells of group-1.



(b) Labels of cells of group-2.

sanjayvidhyadharan.in

Testing RAM Technology and Layout-Related Faults

The problems with the prior memory testing approaches are that DRAMs may be repaired or may have their address lines deliberately scrambled. As a result, consecutive addresses may not be adjacent, so the previously described coupling fault tests will not be effective.

1. Geometry optimisation introducing folding;
2. Address decoder optimisation; ell area optimisation by sharing contacts and well areas;
4. Speed and robustness optimisation based on bitline twisting;
5. Yield optimisation by introducing redundancy
6. Achieving I/O pin compatibility utilising address or data line swap.

A. J. van de Goor and I. Schanstra, "Address and data scrambling: causes and impact on memory tests," *Proceedings First IEEE International Workshop on Electronic Design, Test and Applications '2002*, Christchurch, New Zealand, 2002, pp. 128-136, doi: 10.1109/DELTA.2002.994601.

sanjayvidhyadharan.in

Testing RAM Technology and Layout-Related Faults

Dekker performed this analysis [193, 194, 195] and found faults caused by actual defects modeled as broken wires, shorts between wires, missing contacts, extra contacts, and newly-created parasitic transistors. He mapped these defects into the following functional faults:

1. SAF in a memory cell.
2. A *stuck-open fault* (SOF) in a memory cell.
3. A TF in a memory cell.
4. A *state coupling fault* (SCF) between two memory cells.
5. A CFid between two cells
6. A *data retention fault* (DRF), caused by a broken pull-up device, in which the cell loses its contents over time

Algorithm	Actual physical defect fault coverage							Operation count
	SAF	TF	AF	SOF	SCF	CFid	DRF	
IFA-9	All	All	All		All	All	All	$12 \cdot n$ + Delays
IFA-13	All	All	All	All	All	All	All	$16 \cdot n$ + Delays
MARCH G	All	All	All				All	$23 \cdot n$ + Delays
	All linked CFids All TFs linked with CFids or CFins Some CFins linked with CFids							

sanjayvidhyadharan.in

Testing RAM Technology and Layout-Related Faults

Table 9.20: IFA augmented march test summary.

Algorithm	Actual physical defect fault coverage							Operation count
	SAF	TF	AF	SOF	SCF	CFid	DRF	
IFA-9	All	All	All		All	All	All	$12 \cdot n$ + Delays
IFA-13	All	All	All	All	All	All	All	$16 \cdot n$ + Delays
MARCH G	All	All	All				All	$23 \cdot n$ + Delays
	All linked CFids All TFs linked with CFids or CFins Some CFins linked with CFids							

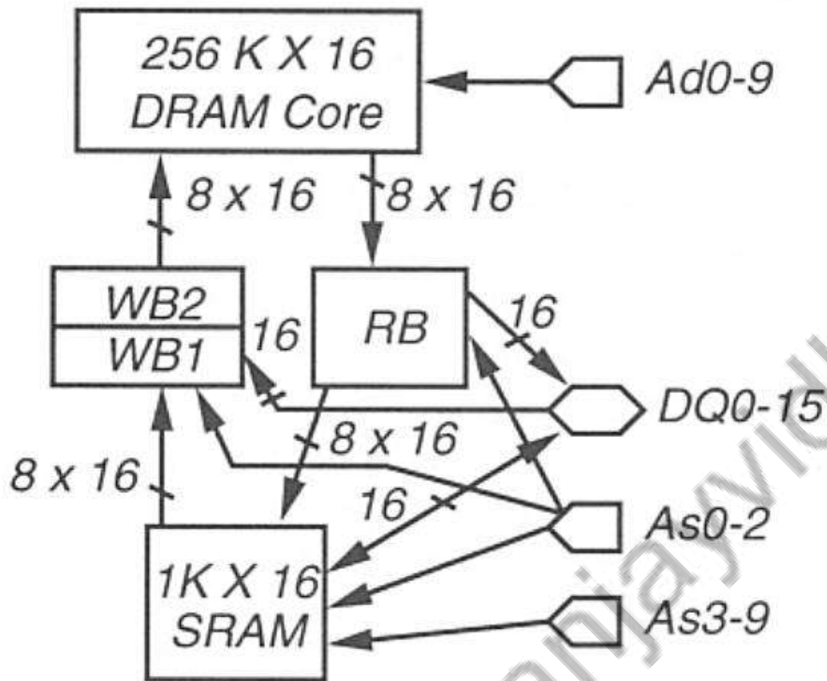
Table 9.21: IFA augmented march test algorithms.

Algorithm	Description	Ref.
IFA-9	$\{ \Uparrow (w0); \Uparrow (r0, w1); \Uparrow (r1, w0); \Downarrow (r0, w1);$ $\Downarrow (r1, w0); Delay; \Uparrow (r0, w1); Delay; \Uparrow (r1) \}$	[193]
IFA-13	$\{ \Uparrow (w0); \Uparrow (r0, w1, r1); \Uparrow (r1, w0, r0); \Downarrow (r0, w1, r1);$ $\Downarrow (r1, w0, r0); Delay; \Uparrow (r0, w1); Delay; \Uparrow (r1) \}$	[193]
MARCH G	$\{ \Updownarrow (w0); \Uparrow (r0, w1, r1, w0, r0, w1);$ $\Uparrow (r1, w0, w1); \Downarrow (r1, w0, w1, w0); \Downarrow (r0, w1, w0);$ $Delay; \Updownarrow (r0, w1, r1); Delay; \Updownarrow (r1, w0, r0) \}$	[532]

Sanjay vidhyadharan.in

Cache RAM Chip Testing

Block diagram of 256K × 16 b Cache DRAM.



1. DRAM Functional Test.
2. SRAM Functional Test
3. Data Transfer Test. (DRAM/SRAM /RB/WB
4. High-Speed Operation Test
5. Concurrent Operation Test
6. Cache Miss Test

Its organization is 256K words × 16 bits of DRAM core, 8 words × 16 bits of *read data buffer* (RB), 8 words × 16 bits of *write data buffer* (WB), and 1K words × 16 bits of SRAM.

sanjayvidhyadharan.in

Functional ROM Chip Testing

ROM testing differs from RAM testing, in that the correct data that the ROM should contain is already known. The SAF model used for ROMs is sometimes a *restricted* SAF model

The preferred ROM testing method is to cycle the ROM through all of its addresses and compress the output bit stream at the ROM outputs using a *linear feedback shift register* (LFSR) in the *automatic test equipment* (ATE).

sanjayvidhyadharani.in

Electrical Parametric Testing

DC Parametric Tests.

1. Voltage Bump Test

Method:	
1.	Zero out memory.
2.	Increase the power supply above V_{CC} in steps of 0.01 V. For each voltage setting, read the memory. Stop as soon as a 1 is read from any location, and record this supply voltage as V_{high}.
3.	Fill the memory with 1s.
4.	Slowly decrease the supply below V_{CC} in steps of 0.01 V. For each setting, read the memory. Stop as soon as a 0 is read from any location, and record this supply voltage as V_{low}.
Possible test outcomes:	
1.	V_{high} and V_{low} are inconsistent with data book values (fails.)

sanjayvidhyadharan.in

Electrical Parametric Testing

DC Parametric Tests.

2. Leakage Test

Method:	
1.	Apply a high to chip select, to deselect the chip.
2.	Apply a high to output enable, and a low to write enable (set the chip pins to be tristated.)
3.	Force a logic high on each data-out line and measure I_{OZ} (output leakage.)
4.	Force a low voltage on each data-out line and measure I_{OZ}.
5.	Select the chip (apply a low to chip select.)
6.	Set the chip in read mode, force a high logic voltage on each address and data-in line, and measure I_I.
7.	Set the chip in read mode, force a low logic voltage on each address and data-in line, and measure I_I.
Possible test outcomes:	
1.	$I_{OZ} < 10\ \mu A$ and $I_I < 10\ \mu A$ (passes.)
2.	$I_{OZ} \geq 10\ \mu A$ (fails.)
3.	$I_I \geq 10\ \mu A$ (fails.)

sanjayvidhyadharan.in

Electrical Parametric Testing

AC Parametric Tests.

1. Address Set-Up Time Sensitivity

Method:	
1.	For every cell: a. Write a 1 in the first memory cell. b. Flip each address bit and write a 0 at the new address. c. Flip each bit back again and read the original cell. d. Repeat Steps a-c with complementary data.
Possible test outcomes:	
1.	The data read from the cell does not match the most recently written data to the cell (fails – excessive address set-up time.)

sanjaywidhyadharan.in

Electrical Parametric Testing

AC Parametric Tests.

2. Access Time Tests

Method:	
1.	Split the memory into two halves.
2.	Write 0s in one half and 1s in the other half.
3.	Read the entire memory and compare it to the expected values.
4.	Alternate between addresses in the two halves.
Possible test outcomes:	
1.	Speed up the read access time until reading fails, and then record the access time delay.

sanjayvidhyadharan.in

Electrical Parametric Testing

AC Parametric Tests.

3. Running Time Tests

Method:	
1.	Perform read operations of 0s and 1s from alternating addresses at a specified rapid speed.
Alternate characterization method:	
1.	Alternate read operations at increasingly rapid speeds until an operation fails.

sanjayvidhyadharan.in

Electrical Parametric Testing

AC Parametric Tests.

4. Tests for Sense Amplifier Recovery Fault. Sense amplifiers can become saturated after reading/writing a long string of identical data values, at which point they are too slow to read the opposite data value

Method:
1. Write the repeating pattern $ddddddd\overline{d}$ to memory locations (d is 0 or 1.)
2. Read a long string of 0s (1s) starting at the first location up to the location with $\overline{d}$.
3. Read a single 1 (0) from the location with the $\overline{d}$.
4. Repeat Steps 2 and 3, but writing rather than reading in Step 2.

sanjayvidhyadharan.in

Electrical Parametric Testing

AC Parametric Tests.

5. Test for Write Recovery Fault. Write recovery faults occur when a write is followed by a read/write at a different address. The two types are *read-after-write* and *write-after-read.*

Method:	
1.	Write a 0 at address a.
2.	Complement all address bits and write a 1 at address $\overline{a}$.
3.	Complement all address bits and read address a (reading a 1 indicates a fault.)
4.	Repeat Steps 1 through 3 for each cell in one-half of the memory.
5.	Repeat Steps 1 through 4 with complementary data.

sanjayvidhyadharan.in

References

1. "Essentials of Electronic Testing, for Digital, Memory and Mixed-Signal VLSI Circuits", Michael L. Bushnell and Vishwani D. Agrawal,–Kluwer Academic Publishers (2000).

2. Video lectures by Professor James Chien-Mo Li
Lab. of Dependable Systems Graduate Institute of Electronics Engineering
National Taiwan University
https://www.youtube.com/watch?v=yfcoKOUV5DM&list=PLvd8d-SyI7hjk_Ci0zpTqImAtpEjdK5JF&index=1

3. NPTEL Lectures
https://www.youtube.com/watch?v=M8VEEaYwlQ&list=PLbMVogVj5nJTClnafWQ9FK2nt3cGG8kCF&index=31

sanjayvidhyadharan.in

Thankyou

sanjayvidhyadharan.in