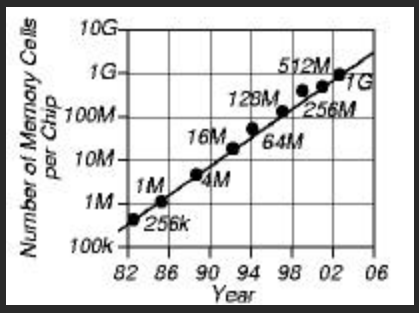
Memory Test

- Memory market and memory complexity
- Notation
- Faults and failures
- MATS+ March Test
- Memory fault models
- March test algorithms
- *Inductive fault analysis*
- Summary

Memory Cells Per Chip



Test Time in Seconds (Memory Size *n Bits*)

Size	Number of Test Algorithm Operations			
n	*n*	*n* X $log_2 n$	$n^{3/2}$	n^2
1 Mb	0.06	1.26	64.5	18.3 hr
4 Mb	0.25	5.54	515.4	293.2 hr
16 Mb	1.01	24.16	1.2 hr	4691.3 hr
64 Mb	4.03	104.7	9.2 hr	75060.0 hr
256 Mb	16.11	451.0	73.3 hr	1200959.9 hr
1 Gb	64.43	1932.8	586.4 hr	19215358.4 hr
2 Gb	128.9	3994.4	1658.6 hr	76861433.7 hr

Notation

- 0 – A cell is in logical state 0
- 1 – A cell is in logical state 1
- X – A cell is in logical state X
- ***A*** – A memory address
- ABF – *AND Bridging Fault*
- AF – *Address Decoder Fault*
- ***B*** – Memory # bits in a word
- BF – *Bridging Fault*
- C – A Memory Cell
- CF – *Coupling Fault*

Notation (Continued)

- CFdyn – *Dynamic Coupling Fault*
- CFid – *Idempotent Coupling Fault*
- CFin – *Inversion Coupling Fault*
- *coupling cell* – cell whose change causes another cell to change
- *coupled cell* – cell forced to change by a coupling cell
- DRF – RAM *Data Retention Fault*
- ***k*** -- Size of a neighborhood
- ***M*** – memory cells, words, or address set
- ***n*** -- # of Memory bits
- ***N*** – Number of address bits: $n = 2^N$
- NPSF – *Neighborhood Pattern Sensitive Fault*

Notation (Continued)

- OBF – *OR Bridging Fault*
- SAF – *Stuck-at Fault*
- SCF – *State Coupling Fault*
- SOAF – *Stuck-Open Address Decoder Fault*
- TF – *Transition Fault*

March Test Notation

- r – Read a memory location
- w – Write a memory location
- r0 – Read a 0 from a memory location
- r1 – Read a 1 from a memory location
- w0 – Write a 0 to a memory location
- w1 – Write a 1 to a memory location
- ↑ – Write a 1 to a cell containing 0
- ↓ – Write a 0 to a cell containing 1

March Test Notation (Continued)

- ⇕ -- Complement the cell contents
- ↑ – Increasing memory addressing
- ↓ – Decreasing memory addressing
- ⇕ – Either increasing or decreasing

More March Test Notation

- ∀ – Any write operation
- < ... > – Denotes a particular fault, ...
- *<I / F >* -- *I* is the fault sensitizing condition, *F* is the faulty cell value
- *<I1, ..., In-1 ; In / F>* – Denotes a fault covering *n* cells
  - *I1, ..., In-1* are fault sensitization conditions in cells 1 through *n - 1* for cell *n*
  - *In* gives sensitization condition for cell *n*
  - If *In* is empty, write *In / F* as *F*

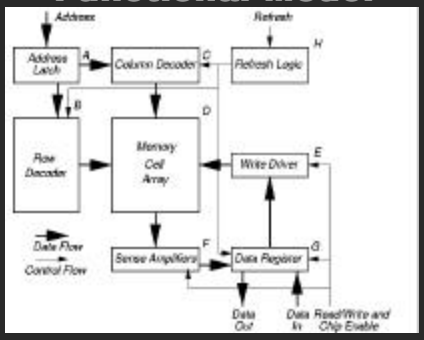
MATS+ March Test

M0: { March element ⇕(w0) }
for cell := 0 to n - 1 (or any other order) do
write 0 to A [cell];
M1: { March element ↑(r0, w1) }
for cell := 0 to n - 1 do
read A [cell]; { Expected value = 0}
write 1 to A [cell];
M2: {March element ↓(r1, w0) }
for cell := n – 1 down to 0 do
read A [cell]; { Expected value = 1 }
write 0 to A [cell];

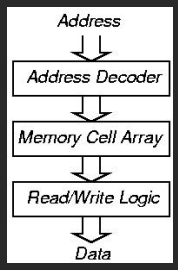
Fault Modeling

- *Behavioral* (black-box) Model – State machine modeling all memory content combinations – Intractable
- *Functional* (gray-box) Model – Used
- *Logic Gate* Model – Not used Inadequately models transistors & capacitors
- *Electrical* Model – Very expensive
- *Geometrical* Model – Layout Model
  - Used with *Inductive Fault Analysis*

Functional Model



Simplified Functional Model



Subset Functional Faults

	Functional fault
a	Cell stuck
b	Driver stuck
c	Read/write line stuck
d	Chip-select line stuck
e	Data line stuck
f	Open circuit in data line
g	Short circuit between data lines
h	Crosstalk between data lines

Subset Functional Faults (Continued)

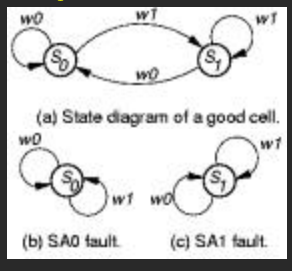
	Functional fault
i	Address line stuck
j	Open circuit in address line
k	Shorts between address lines
l	Open circuit in decoder
m	Wrong address access
n	Multiple simultaneous address access
o	Cell can be set to 0 but not to 1 (or vice versa)
p	Pattern sensitive cell interaction

Reduced Functional Faults

	Fault
SAF	Stuck-at fault
TF	Transition fault
CF	Coupling fault
NPSF	Neighborhood Pattern Sensitive fault

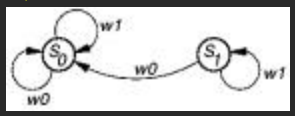
Stuck-at Faults

- ***Condition*: For each cell, must read a 0 and a 1.**
- **<∀/0> (<∀/1>)**



Transition Faults

- **Cell fails to make 0→1 or 1→0 transition**
- ***Condition*: Each cell must undergo a ↑ transition and a ↓ transition, and be read after such, before undergoing any further transitions.**
- **<↑/0>, <↓/1>**



<↑/0> transition fault

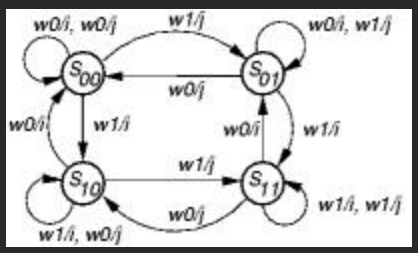
Coupling Faults

- ***Coupling Fault*** **(CF): Transition in bit *j* causes unwanted change in bit *i***
- ***2-Coupling Fault*****: Involves 2 cells, special case of *k-Coupling Fault***
  - **Must restrict *k* cells to make practical**
- ***Inversion*** **and *Idempotent* CFs – special cases of *2-Coupling Faults***
- ***Bridging*** **and *State Coupling Faults* involve any # of cells, caused by logic level**
- ***Dynamic Coupling Fault*** **(CFdyn) – Read or write on *j* forces *i* to 0 or 1**

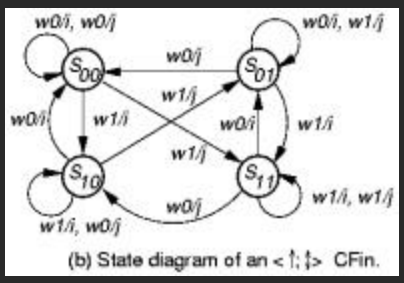
Inversion Coupling Faults (CFin)

- **↑ or ↓ in cell *j* inverts contents of cell *i***
- ***Condition*****: For all cells that are coupled, each should be read after a series of possible CFins may have occurred, and the # of coupled cell transitions must be odd (to prevent the CFins from masking each other).**
- **<↑;↕> and <↓;↕>**

Good Machine State Transition Diagram



CFin State Transition Diagram

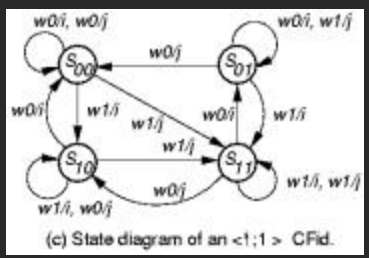


(b) State diagram of an <↑;↕> CFin.

Idempotent Coupling Faults (CFid)

- **↑ or ↓ transition in *j* sets cell *i* to 0 or 1**
- ***Condition*****: For all coupled faults, each should be read after a series of possible CFids may have happened, such that the sensitized CFids do not mask each other.**
- ***Asymmetric*****: coupled cell only does ↑ or ↓**
- ***Symmetric*****: coupled cell does both due to fault**
- **<↑; 0>, <↑; 1>, <↓; 0>, <↓; 1>**

CFid Example



(c) State diagram of an <1;1> CFid.

Dynamic Coupling Faults (CFdyn)

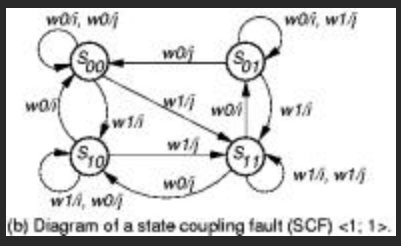
- Read or write in cell of 1 word forces cell in different word to 0 or 1
- <r0 | w0 ; 0>, <r0 | w0 ; 1>, < r1 | w1 ; 0>, and <r1 | w1; 1>
  - | Denotes "OR" of two operations
- More general than CFid, because a CFdyn can be sensitized by any read or write operation

Bridging Faults

- Short circuit between 2+ cells or lines
- 0 or 1 state of *coupling cell*, rather than coupling cell transition, causes *coupled cell* change
- Bidirectional fault – *i* affects *j*, *j* affects *i*
- *AND Bridging Faults* (ABF):
  - < 0,0 / 0,0 >, <0,1 / 0,0 >, <1,0 / 0,0>, <1,1 / 1,1>
- *OR Bridging Faults* (OBF):
  - < 0,0 / 0,0 >, <0,1 / 1,1 >, <1,0 / 1,1>, <1,1 / 1,1>

State Coupling Faults

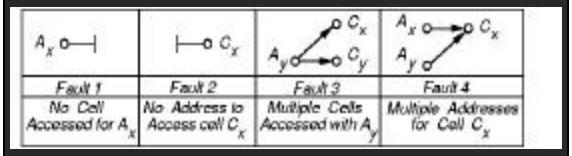
- *Coupling cell / line j* is in a given state *y* that forces *coupled cell / line i* into state *x*
- < 0;0 >, < 0;1 >, < 1;0 >, < 1;1 >

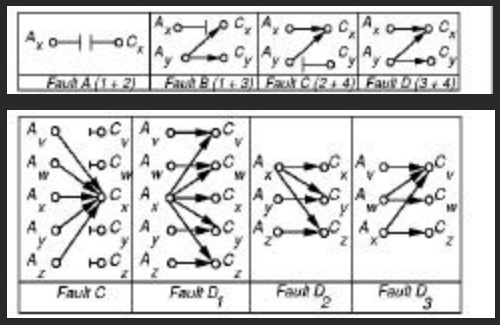


(b) Diagram of a state coupling fault (SCF) <1; 1>.

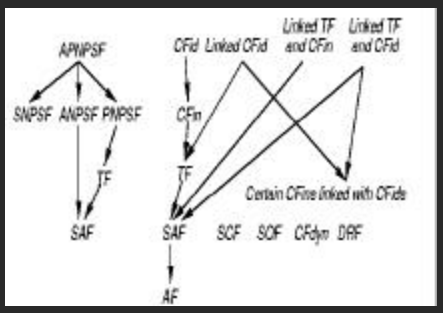
Address Decoder Faults (ADFs)

- Address decoding error assumptions:
  - Decoder does not become sequential
  - Same behavior during both read & write
- Multiple ADFs must be tested for
- Decoders have CMOS stuck-open faults





Fault Hierarchy



Functional RAM Testing with March Tests

- March Tests can detect AFs – NPSF Tests Cannot
- Conditions for AF detection:
  - Need ↑(r x, w $\bar{x}$)
  - Need ↓(r $\bar{x}$, w x)
- In the following March tests, addressing orders can be interchanged

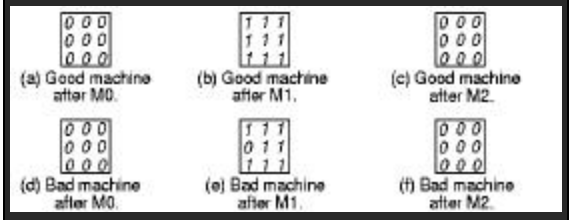
Irredundant March Tests

Algorithm	Description
MATS	{ ⇕(w0); ⇕(r0, w1); ⇕(r1) }
MATS+	{ ⇕(w0); ↑(r0, w1); ↓(r1, w0) }
MATS++	{ ⇕(w0); ↑(r0, w1); ↓(r1, w0, r0) }
MARCH X	{ ⇕(w0); ↑(r0, w1); ↓(r1, w0); ⇕(r0) }
MARCH C—	{ ⇕(w0); ↑(r0, w1); ↑(r1, w0); ↓(r0, w1); ↓(r1, w0); ⇕(r0) }
MARCH A	{ ⇕(w0); ↑(r0, w1, w0, w1); ↑(r1, w0, w1); ↓(r1, w0, w1, w0); ↓(r0, w1, w0) }
MARCH Y	{ ⇕(w0); ↑(r0, w1, r1); ↓(r1, w0, r0); ⇕(r0) }
MARCH B	{ ⇕(w0); ↑(r0, w1, r1, w0, r0, w1); ↑(r1, w0, w1); ↓(r1, w0, w1, w0); ↓(r0, w1, w0) }

March Test Complexity

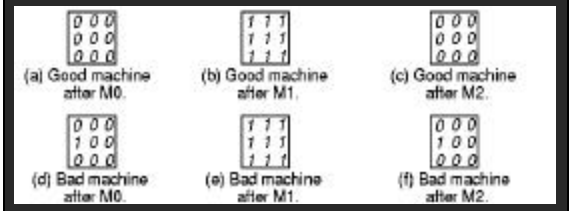
Algorithm	Complexity
MATS	4*n*
MATS+	5*n*
MATS++	6*n*
MARCH X	6*n*
MARCH C—	10*n*
MARCH A	15*n*
MARCH Y	8*n*
MARCH B	17*n*

MATS+ Example Cell (2,1) SA0 Fault



MATS+:
{ M0: ⇕ (w0); M1: ↑(r0, w1); M2: ↓(r1, w0) }

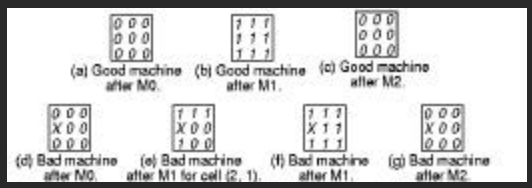
MATS+ Example Cell (2, 1) SA1 Fault



MATS+:
{ M0: ⇕ (w0); M1: ↑(r0, w1); M2: ↓(r1, w0) }

MATS+ Example Multiple AF Type C

- Cell (2,1) is not addressable
- Address (2,1) maps into (3,1) & vice versa
- Can't write (2,1), read (2,1) gives random #



MATS+:
{ M0: ⇕ (w0); M1: ↑(r0, w1); M2: ↓(r1), w0 }

Pattern Sensitive and Electrical Memory Test

- Notation
- Neighborhood pattern sensitive fault algorithms
- Cache DRAM and ROM tests
- Memory Electrical Parametric Tests
- Summary

Notation

- ANPSF – *Active Neighborhood Pattern Sensitive Fault*
- APNPSF – *Active and Passive Neighborhood PSF*
- *Neighborhood* – Immediate cluster of cells whose pattern makes base cell fail
- NPSF – *Neighborhood Pattern Sensitive Fault*
- PNPSF -- *Passive Neighborhood PSF*
- SNPSF -- *Static Neighborhood Pattern Sensitive Fault*

Neighborhood Pattern Sensitive Coupling Faults

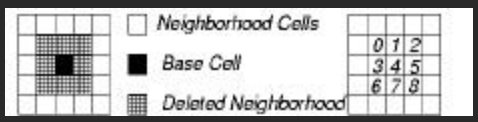
- Cell *i*'s ability to change influenced by all other memory cell contents, which may be a 0/1 pattern or a transition pattern.
- Most general *k-Coupling Fault*
- *Base cell* -- cell under test
- *Deleted neighborhood* – neighborhood without the base cell
- Neighborhood is single position around base cell
- Testing assumes read operations are fault free

Type 1 Active NPSF

- *Active*: Base cell changes when one deleted neighborhood cell transitions
- *Condition for detection & location*: Each base cell must be read in state 0 and state 1, for all possible deleted neighborhood pattern changes.
- $C_{i,j}$ <d_0, d_1, d_3, d_4 ; b>
- $C_{i,j}$ <0, ↓, 1, 1; 0> and $C_{i,j}$ <0, ↓, 1, 1; ↑>

Type 2 Active NPSF

- Used when diagonal couplings are significant, and do not necessarily cause horizontal/vertical coupling



Passive NPSF

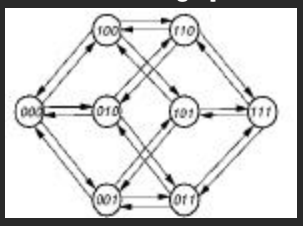
- *Passive*: A certain neighborhood pattern prevents the base cell from changing
- *Condition for detection and location*: Each base cell must be written and read in state 0 and in state 1, for all deleted neighborhood pattern changes.
- ↑/0 (↓/1) – Base cell fault effect indicating that base cannot change

Static NPSF

- ***Static*: Base cell forced into a particular state when deleted neighborhood contains particular pattern.**
- **Differs from *active* – need not have a transition to sensitive SNPSF**
- ***Condition for detection and location*: Apply all 0 and 1 combinations to *k*-cell neighborhood, and verify that each base cell was written.**
- **$C_{i,j}$ < 0, 1, 0, 1; - / 0> and $C_{i,j}$ < 0, 1, 0, 1; - / 1>**

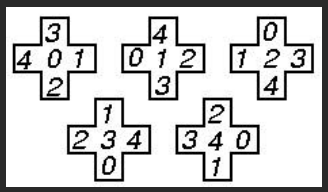
Eulerian / Hamiltonian Graph Tour Sequences

- **Both used for writing shorter patterns**
- ***Hamiltonian* – traverses each graph node once**
- ***Eulerian* – traverses each graph arc exactly once**



Type 1 Tiling Neighborhoods

- **Write changes *k* different neighborhoods**
- ***Tiling* Method: Cover all memory with non-overlapping neighborhoods**



NPSF Fault Detection and Location Algorithm

1. **write base-cells with 0;**
2. **loop**
   apply a pattern; { it could change the base-cell from 0 to 1. }
   read base-cell;
   endloop;
3. **write base-cells with 1;**
4. **loop**
   apply a pattern; { it could change the base-cell from 1 to 0. }
   read base-cell;
   endloop;

NPSF Testing Algorithm Summary

- **A: *active*, P: *passive*, S: *static***
- **D: *Detects Faults*, L: *Locates Faults***

Algorithm	Fault Location?	Fault Coverage					Operation Count
				NPSF			
		SAF	TF	A	P	S	
TDANPSF1G	No	L		D			163.5 *n*
TLAPNPSF1G	Yes	L	L	L	L	L	195.5 *n*
TLAPNPSF2T	Yes	L	L	L	L		5122 *n*
TLAPNPSF1T	Yes	L	L	L	L		194 *n*
TLSNPSF1G	Yes	L				L	43.5 *n*
TLSNPSF1T	Yes	L				L	39.2 *n*
TLSNPSF2T	Yes	L				L	569.78 *n*
TDSNPSF1G	No	L				D	36.125 *n*

Fault Hierarchy