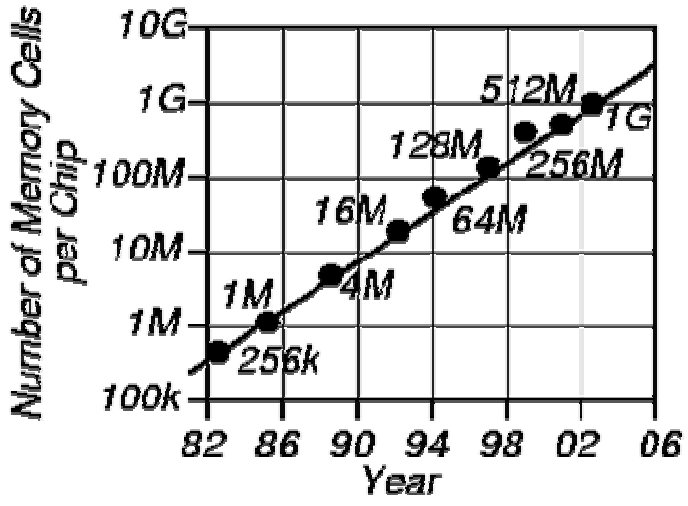
Memory Testing

Memory Cells Per Chip



Test Time in Seconds
(Memory Size n Bits, Memory Cycle Time 60ns)

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

3

Fault Types

- Permanent
  - System is broken and stays broken the same way indefinitely
- Transient
  - Fault temporarily affects the system behavior, and then the system reverts to the good machine
  - Time dependency, caused by environmental condition
- Intermittent
  - Sometimes causes a failure, sometimes does not

4

Failure Mechanisms

- **Permanent faults:**
  - **Missing/Added Electrical Connection**
  - **Broken Component (IC mask defect or silicon-to-metal connection)**
  - **Burnt-out Chip Wire**
  - **Corroded connection between chip & package**
  - **Chip logic error (Pentium division bug)**

5

- **Transient Faults:**
  - **Cosmic Ray**
  - **An a particle (ionized Helium atom)**
  - **Air pollution (causes wire short/open)**
  - **Humidity (temporary short)**
  - **Temperature (temporary logic error)**
  - **Pressure (temporary wire open/short)**
  - **Vibration (temporary wire open)**
  - **Power Supply Fluctuation (logic error)**
  - **Electromagnetic Interference (coupling)**
  - **Static Electrical Discharge (change state)**
  - **Ground Loop (misinterpreted logic value)**

6

- **Intermittent Faults:**
  - Loose Connections
  - Aging Components (changed logic delays)
  - Hazards and Races in critical timing paths (bad design)
  - Resistor, Capacitor, Inductor variances (timing faults)
  - Physical Irregularities (narrow wire -- high resistance)
  - Electrical Noise (memory state changes)

7

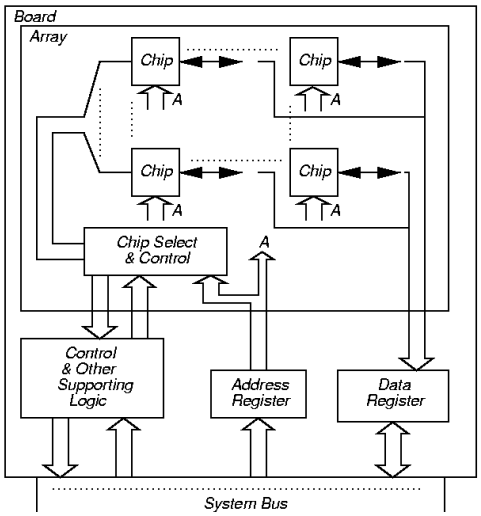
Physical Failure Mechanisms

- Corrosion
- Electromigration
- Bonding Deterioration
  - Au package wires interdiffuse with Al chip pads
- Ionic Contamination
  - Na^+ diffuses through package and into FET gate oxide
- Alloying
  - Al migrates from metal layers into Si substrate
- Radiation and Cosmic Rays
  - 8 MeV, collides with Si lattice, generates n-p pairs, causes soft memory error

8

Memory Test Levels

Chip, Array, & Board



9

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

10

⇕ -- **Complement the cell contents**

⇑ -- **Increasing memory addressing**

⇓ -- **Decreasing memory addressing**

⇕ -- **Either increasing or decreasing**

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];

O(n) complexity

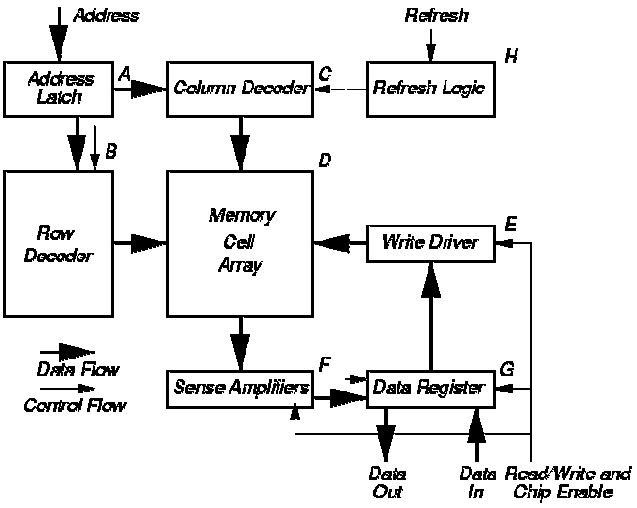
NPSFs cannot be detected

Fault Modeling

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

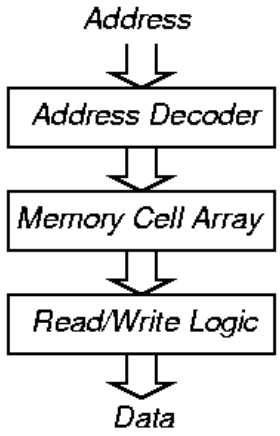
13

Detailed Functional Model



14

Simplified Functional Model



15

Functional Faults

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

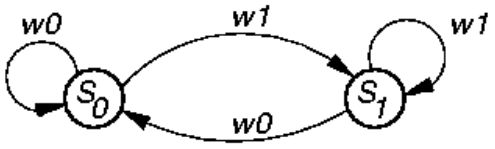
16

Stuck-at Faults

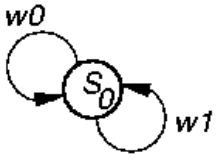
- **Manifestation:**
  - **The logic value of a cell or line is always at 0 or 1.**
- **Necessary condition for detection:**
  - **For each cell, a 0 and a 1 must be read.**

17

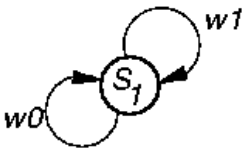
Stuck-at Faults (contd.)



(a) State diagram of a good cell.



(b) SA0 fault.



(c) SA1 fault.

18

Transition Faults

- It is a special case of stuck-at fault, in which a cell fails to make a 0 → 1, or a 1 → 0 transition.
- Up transition fault:
  - Denoted as: $< \uparrow / 0 >$
- Down transition fault:
  - Denoted as: $< \downarrow / 1 >$

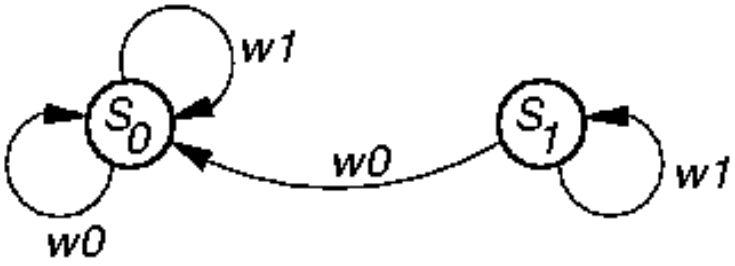
19

Transition Faults (contd.)

- Necessary condition to detect and locate all transition faults:
  - Each cell must undergo a $\uparrow$ transition and a $\downarrow$ transition, and be read after each, before undergoing any further transitions.

20

Transition Faults (contd.)



Up transition fault $< \uparrow / 0 >$

21

Coupling Faults

- Coupling Fault (CF):
  - Transition in memory bit j causes unwanted change in memory bit i.
- 2-Coupling Fault:
  - Involves 2 cells; special case of k-coupling fault.
  - Must restrict k to make the fault model practical.
    - For example, NPSF (to be discussed later).
- Inversion and Idempotent CF's
  - Special cases of 2-coupling faults.

22

Inversion Coupling Faults (CFin)

- A ↑ or ↓ transition in cell j inverts the contents of cell i.
  - Cell i is said to be *coupled* to cell j.
- Two possible CFin's are:
  - < ↑ ; ↕ > and < ↓ ; ↕>
- Necessary condition for detection:
  - For all cells that are coupled, each should be read after a series of possible CFin's may have occurred (due to writing into the coupling cells), and the number of coupled cell transitions must be odd (to prevent the CFin's from masking each other).

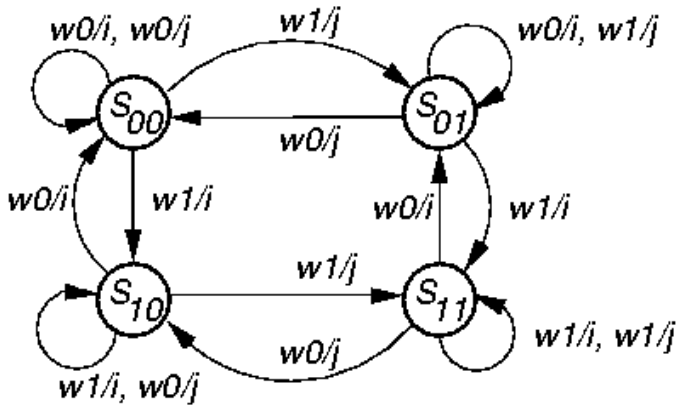
23

CFin (contd.)

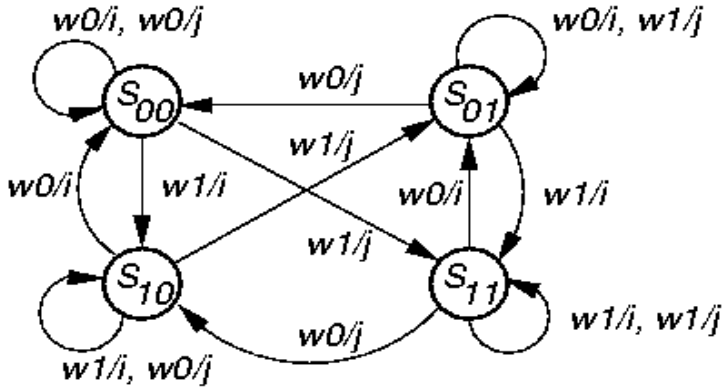
- Theorem:
  - Not all linked CFin's are detected by March tests.

24

CFin: Good Machine State



CFin: Faulty Machine State



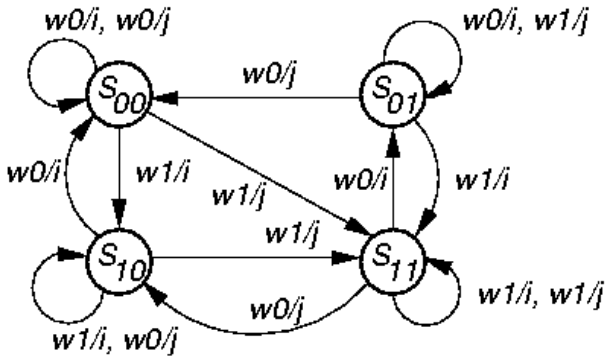
(b) State diagram of an < ↑; ↕> CFin.

Idempotent Coupling Faults (CFid)

- A ↑ or ↓ transition in cell j sets cell i to 0 or 1.
  - Denoted as <↑; 0>, <↑;1>, <↓;0>, <↓;1>
- Necessary condition for detection:
  - For all cells that are coupled, each should be read after a series of possible CFid's may have occurred (due to writing into the coupling cells), such that the sensitized CFid's do not mask each other.
- Asymmetric CFid:
  - Coupled cell only does ↑ or ↓.
- SymmetricCFid:
  - Coupled cell does both due to fault.

27

CFid: Faulty Machine State



(c) State diagram of an <↑;1 > CFid.

28

Dynamic Coupling Faults (CFdyn)

- Read or write in cell of 1 word forces cell in a 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.

29

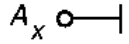
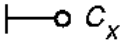
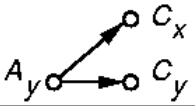
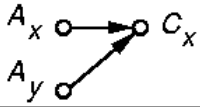
Bridging Faults

- Short circuit between 2 or more 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>

30

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 1	*Fault 2*	*Fault 3*	*Fault 4*
No Cell Accessed for A_x	*No Address to Access cell C_x*	*Multiple Cells Accessed with A_y*	*Multiple Addresses for Cell C_x*

31

- **Theorem:**
  - **A March test satisfying the following conditions detects all address decoder faults:**

    a) ⇕ (rx, ……, wx)

    b) ⇕ (rx, ……, wx)

    where the dots indicate any number of read or write operations.

32

Reduced Functional Faults

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 (1) but not to 1 (0)
NPSF	*p*	Pattern sensitive cell interaction

Functional RAM Testing with March Tests

- March Tests can detect AFs -- NPSF Tests Cannot
- Conditions for AF detection:
  - It must read the value x from cell 0, write x' to cell 0; read x from cell 1, write x' to cell 1; for the entire memory.
  - It must read the value x' from cell n-1, write x to cell n-1; read x' from cell n-2, write x to cell n-2; for the entire memory.
- 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) }

35

Irredundant March Test Summary

Algorithm	SAF	AF	TF	CF in	CF id	CF dyn	Linked Faults
MATS	All	Some					
MATS+	All	All					
MATS++	All	All	All				
MARCH X	All	All	All	All			
MARCH C--	All	All	All	All	All	All	
MARCH A	All	All	All	All			Some
MARCH Y	All	All	All	All			Some
MARCH B	All	All	All	All			Some

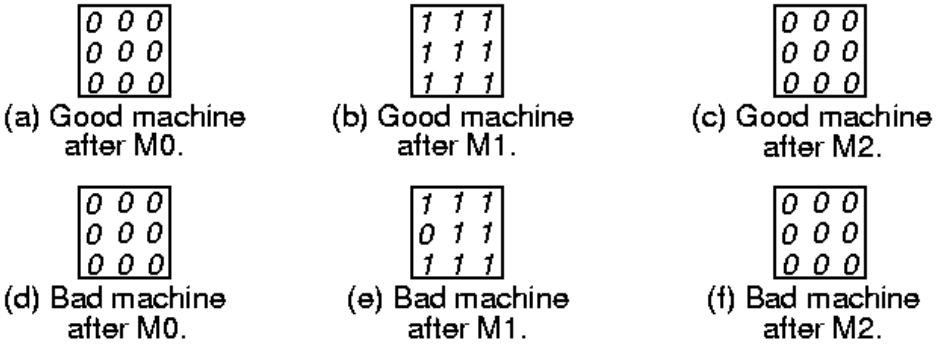
36

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*

37

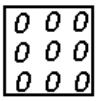
MATS+ Example:: Cell (2,1) SA0 Fault



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

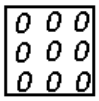
38

MATS+ Example:: Cell (2, 1) SA1 Fault

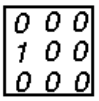


(a) Good machine after M0.

(b) Good machine after M1.

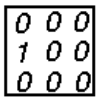


(c) Good machine after M2.



(d) Bad machine after M0.

(e) Bad machine after M1.



(f) Bad machine after M2.

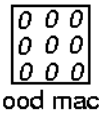
MATS+:
{ M0: ⇕(w0); M1: ⇑(r0, w1); M2: ⇓(r1, w0) }

MATS+ Example:: Multiple AF

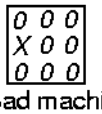
- Cell (2,1) is not addressable
- Address (2,1) maps into (3,1) & vice versa
- Can't write (2,1), read (2,1) gives random #

(a) Good machine after M0.

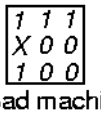
(b) Good machine after M1.



(c) Good machine after M2.

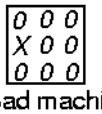


(d) Bad machine after M0.



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

(f) Bad machine after M1.



(g) Bad machine after M2.

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

Pattern Sensitive Faults

- Base Cell – cell under test
- Neighborhood -- Immediate cluster of cells whose pattern makes base cell fail
- Deleted Neighborhood – Neighborhood without the base cell
- NPSF -- Neighborhood Pattern Sensitive Fault (Most General k-coupling fault)
  - ANPSF -- Active Neighborhood Pattern Sensitive Fault
  - PNPSF -- Passive Neighborhood PSF
  - SNPSF -- Static Neighborhood Pattern Sensitive Fault

41

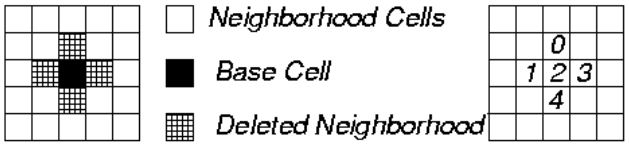
Neighborhood Pattern Sensitive Coupling Faults

- Cell i's ability to change influenced by all other memory cell contents, which may be a 0/1 pattern or a transition pattern.
- Testing assumes read operations are fault free

42

Type 1 Active NPSF

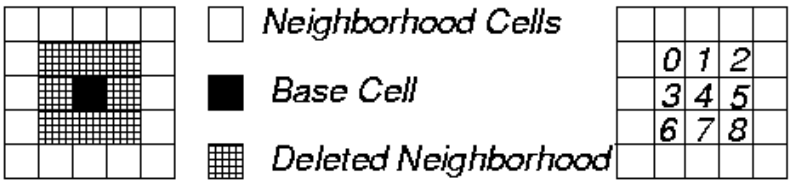
- ***Active***: **Base cell changes when one deleted neighborhood cell transitions**
- ***Condition for detection & location***: **Each base cell must be read in state 0 and state 1, for all possible deleted neighborhood pattern changes.**
- **$C_{i,j} <d_0, d_1, d_3, d_4 ; b>$**
- **$C_{i,j} <0, \downarrow,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



43

Type 2 Active NPSF

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



44

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**

45

Static NPSF

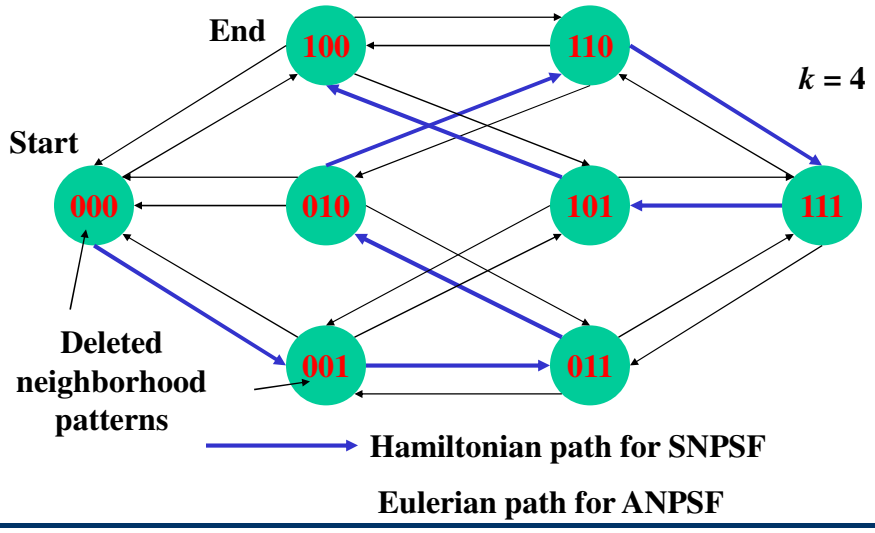
- **Static:**
  - **Base cell forced into a particular state when deleted neighborhood contains particular pattern.**
- **Differs from active -- need not have a transition to sensitive SNPSF**
- **Condition for detection and location:**
  - **Apply all 0 and 1 combinations to k-cell neighborhood, and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

46

Number of NPSFs

- Active Neighborhood Pattern Sensitive Faults (ANPSF)
  - Base cell 0 and 1
    - ↑ and ↓ transitions in $k-1$ cells
      - All 0-1 patterns in $k-2$ cells
  - $2(k-1)\ 2 \times 2^{k-2} = (k-1)\ 2^k$ patterns
- Passive Neighborhood Pattern Sensitive Faults (PNPSF)
  - Base cell ↑ and ↓ transition
    - All 0-1 patterns in $k-1$ cells
  - $2 \times 2^{k-1} = 2^k$ patterns
- Total APNPSF patterns = $(k-1)\ 2^k + 2^k = k\ 2^k$
- Static Neighborhood Patterns (SNP) = 2^k

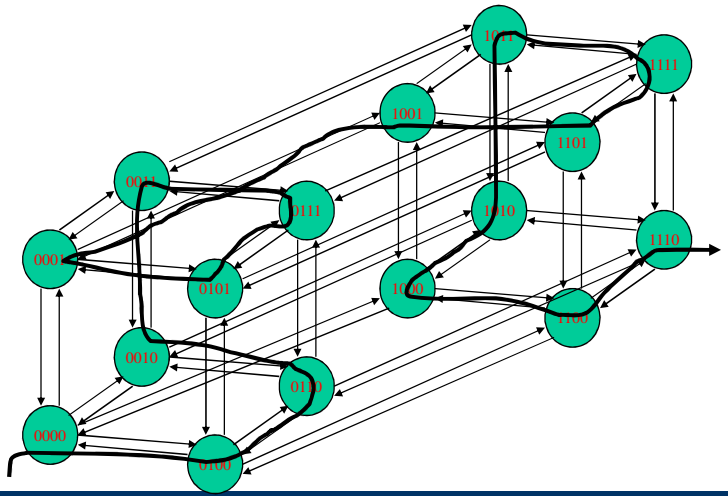
Sequencing Neighborhood Patterns with Minimal Writes



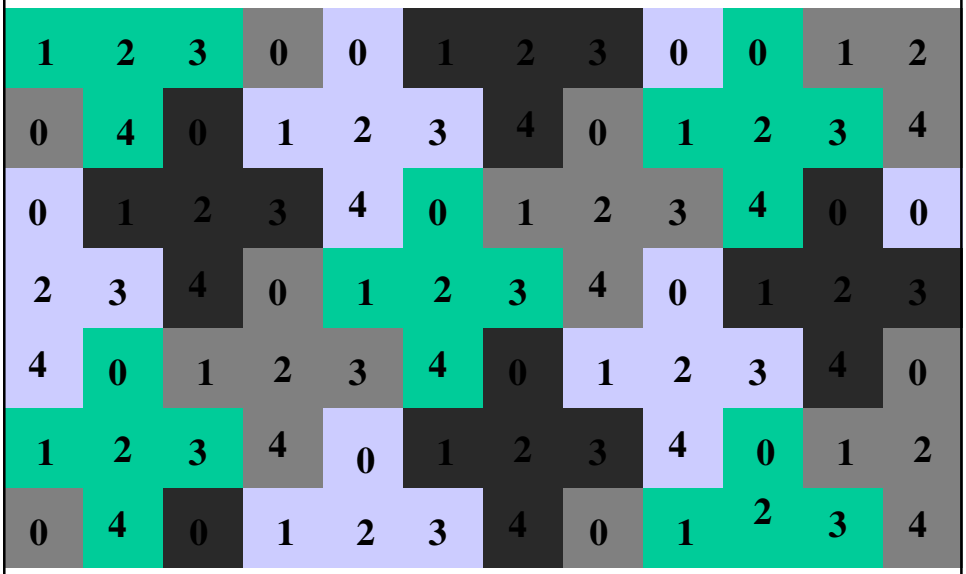
- **Euler Path:**
  - It is a path in a graph that traverses every edge of the graph exactly once.
- **Hamiltonian Path:**
  - It is a path that starts with some vertex of a graph, and traverses every vertex exactly once.

49

Hamiltonian Path, *k* = 5



Tiling for Type-1 Neighborhood



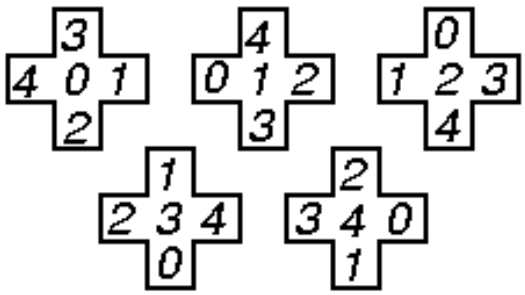
Total Patterns for *n* Cells

Fault type	Without tiling	With tiling
Static neighborhood patterns sensitive faults (SNPSF)	$n \times 2^k$	$n \times 2^k / k$
Active and passive neighborhood pattern sensitive faults (APNPSF)	$n \times k \times 2^k$	$n \times k \times 2^k / k$

k = neighborhood size = 5 or 9

Type 1 Tiling Neighborhoods

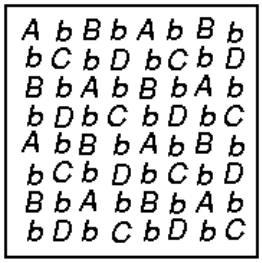
- Write changes *k* different neighborhoods.
- *Tiling* Method: Cover all memory with non-overlapping neighborhoods.



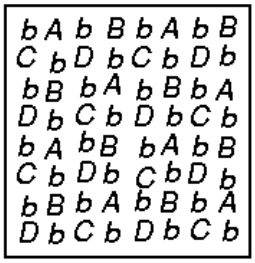
53

Two Group Method

- Only for Type-1 neighborhoods.
- Use checkerboard pattern, cell is simultaneously a base cell in group 1, and a deleted neighborhood cell in 2.



(a) Labels of cells of group-1.



(b) Labels of cells of group-2.

54

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;

55

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*

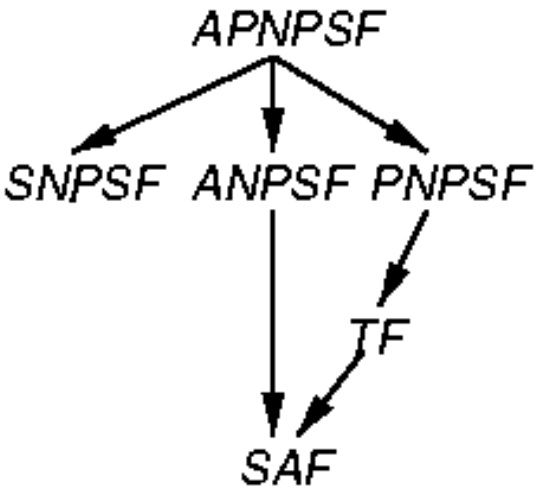
56

NPSF Testing Algorithms

Algorithm	Neigh-bor-hood	Method	*k*
TDANPSF1G	Type-1	2 Group	5
TLAPNPSF1G	Type-1	2 Group	5
TLAPNPSF2T	Type-2	Tiling	9
TLAPNPSF1T	Type-1	Tiling	5
TLSNPSF1G	Type-1	2 Group	5
TLSNPSF1T	Type-1	Tiling	5
TLSNPSF2T	Type-2	Tiling	9
TDSNPSF1G	Type-1	2 Group	5

57

Fault Hierarchy



58

Memory Testing Summary

- **Multiple fault models are essential**
- **Combination of tests is essential:**
  - **March -- SRAM and DRAM**
  - **NPSF -- DRAM**
  - **DC Parametric -- Both**
  - **AC Parametric -- Both**