VLSI Design Verification and Testing

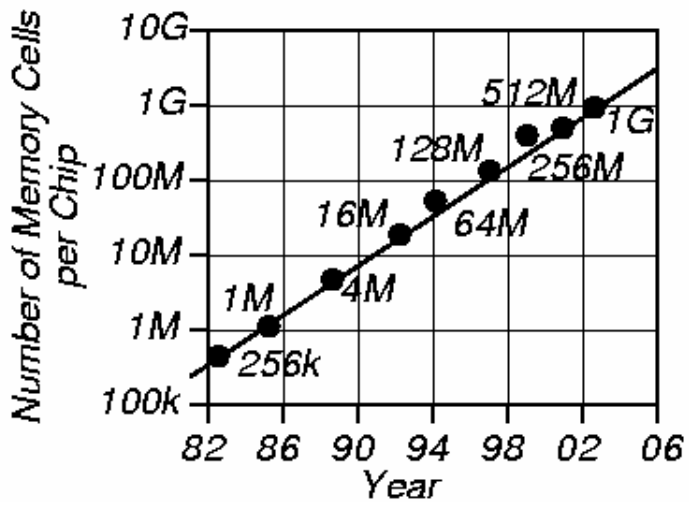
Memory Testing

Mohammad Tehranipoor
Electrical and Computer Engineering
University of Connecticut

Overview

- Motivation and introduction
- Functional model of a memory
- A simple minded test and its limitations
- Fault models
- March tests and their capabilities
- Neighborhood tests
- 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

Memory cycle time = 60ns

Faults

- ***System*** **-- Mixed electronic, electromechanical, chemical, and photonic system (MEMS technology)**
- ***Failure*** **-- Incorrect or interrupted system behavior**
- ***Error*** **-- Manifestation of fault in system**
- ***Fault*** **-- Physical difference between good & bad system behavior**

Fault Types

- **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**

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)**

Failure Mechanisms (Continued)

- **Transient Faults:**
  - **Cosmic Ray**
  - **An α 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)**

Failure Mechanisms (Continued)

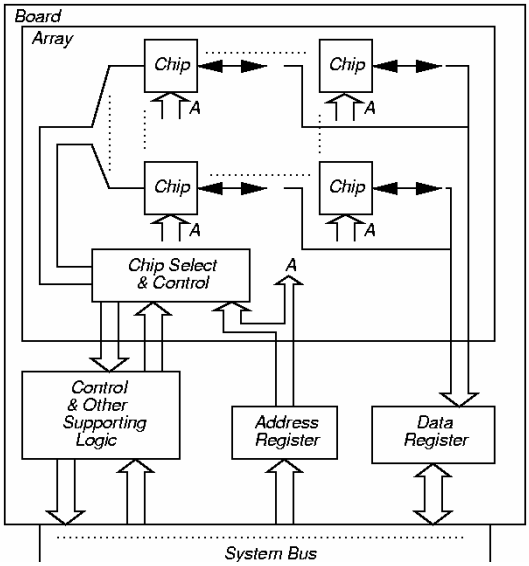
- **Intermittent Faults:**
  - **Loose Connections**
  - **Aging Components (changed logic delays)**
  - **Hazards and Races in critical timing paths (bad design)**
  - **Resistor, Capacitor, Inductor variances (timing faults)**
  - **Physical Irregularities (narrow wire -- high resistance)**
  - **Electrical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

Fault Modeling

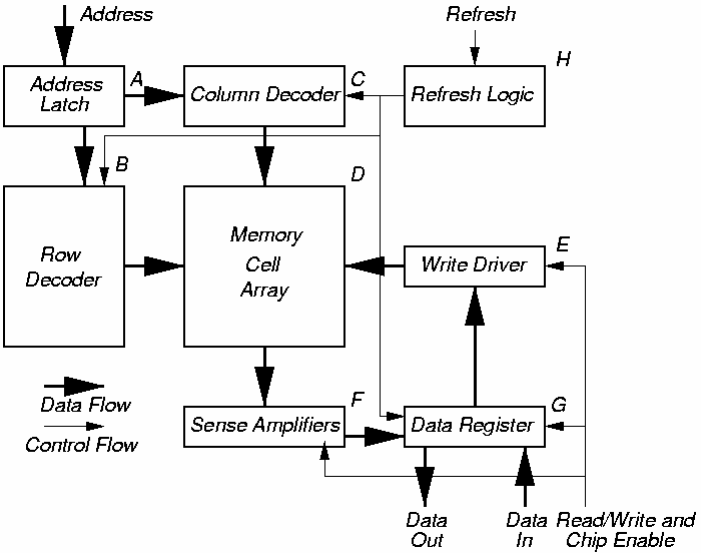
- ***Behavioral*** **(black-box) Model -- State machine modeling all memory content combinations -- Intractable**
- ***Functional*** **(gray-box) Model -- Used**
- ***Logic Gate*** **Model -- Not used Inadequately models transistors & capacitors**
- ***Electrical*** **Model -- Very expensive**
- ***Geometrical***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

Memory Test Levels

Chip, Array, & Board



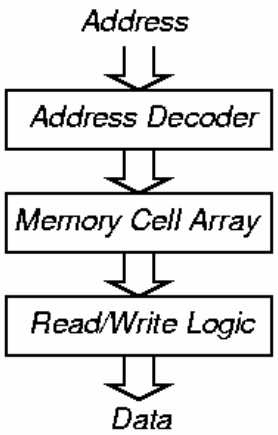
Functional Model



Reduced Functional Model (van de Goor)

- ***n*** **Memory bits,** ***B*** **bits/word,** ***n*****/*****B*** **addresses**
- **Access happens when Address Latch contents change**
- **Low-order address bits operate column decoder, high-order operate row decoder**
- **read -- Precharge bit lines, then activate row**
- **write -- Keep driving bit lines during evaluation**
- **Refresh -- Read all bits in 1 row and simultaneously refresh them**

Simplified Functional Model



A simple minded test

```
for cell := 0 to n - 1 (or any other order) do
    write 0 to A [cell];
    read A [cell]; { Expected value = 0}
    write 1 to A [cell];
    read A [cell]; { Expected value = 1 }
    end for;
```

What does this test achieve?

What kind of faults does it detect and its fault coverage?

Functional Faults

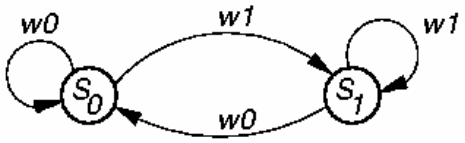
Fault		Functional fault
SAF	*a*	Cell stuck
SAF	*b*	Driver stuck
SAF	*c*	Read/write line stuck
SAF	*d*	Chip-select line stuck
SAF	*e*	Data line stuck
SAF	*f*	Open circuit in data line
CF	*g*	Short circuit between data lines
CF	*h*	Crosstalk between data lines
AF	*i*	Address line stuck
AF	*j*	Open circuit 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

Reduced Functional Faults

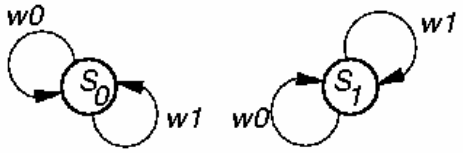
	Fault
SAF	**Stuck-at fault**
TF	**Transition fault**
CF	**Coupling fault**
NPSF	**Neighborhood Pattern Sensitive fault**

Stuck-at Faults

- ***Condition*: For each cell, must read a 0 and a 1.**



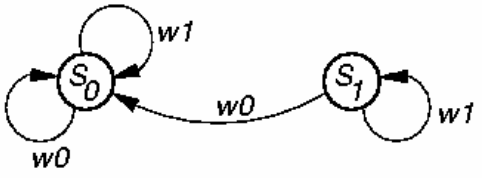
(a) State diagram of a good cell.



(b) SA0 fault. (c) SA1 fault.

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> transition fault

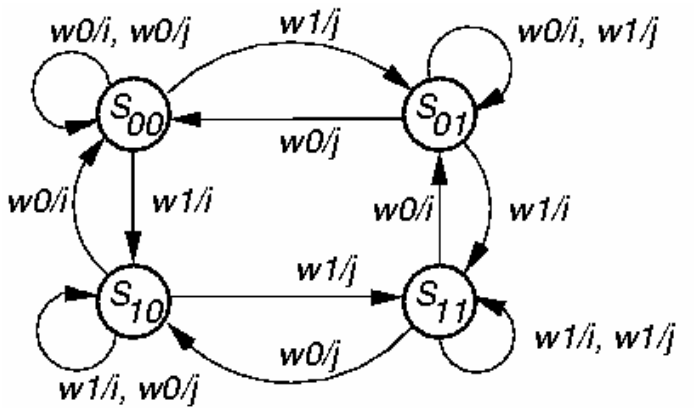
Coupling Faults

- *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

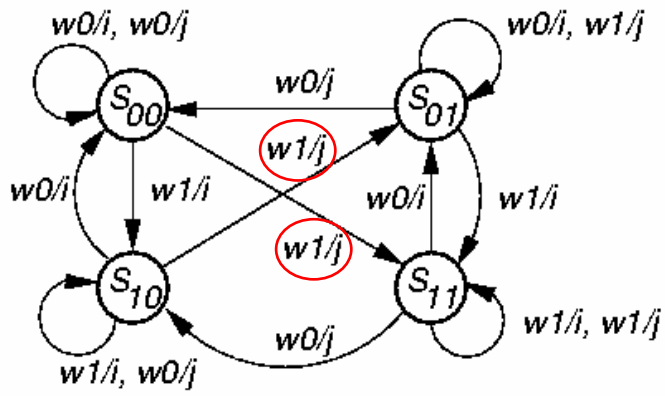
Inversion Coupling Faults (CFin)

- ↑ 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

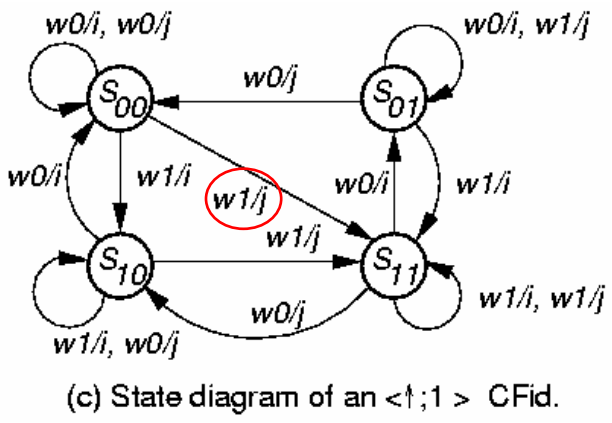


(b) State diagram of an < ↑; ↕> CFin.

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 > CFid.

Dynamic Coupling Faults (CFdyn)

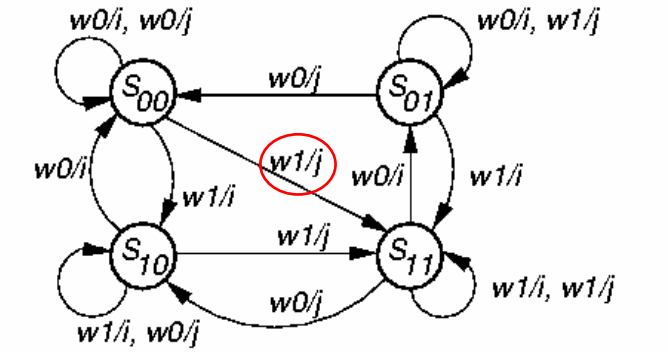
- **Read or write in cell of 1 word forces cell in different word to 0 or 1**
- **<r0 | w0 ; 0>, <r0 | w0 ; 1>, < r1 | w1 ; 0>, and <r1 | w1; 1>**
  - **| Denotes "OR" of two operations**
- **More general than CFid, because a CFdyn can be sensitized by any read or write operation**

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>**

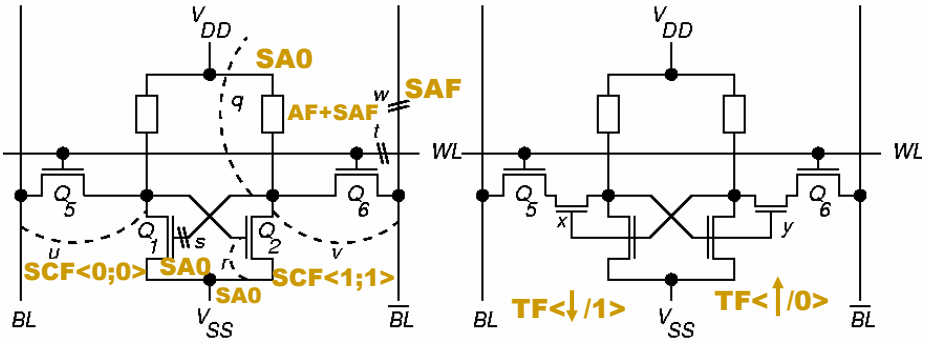
State Coupling Faults

- ***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 >**

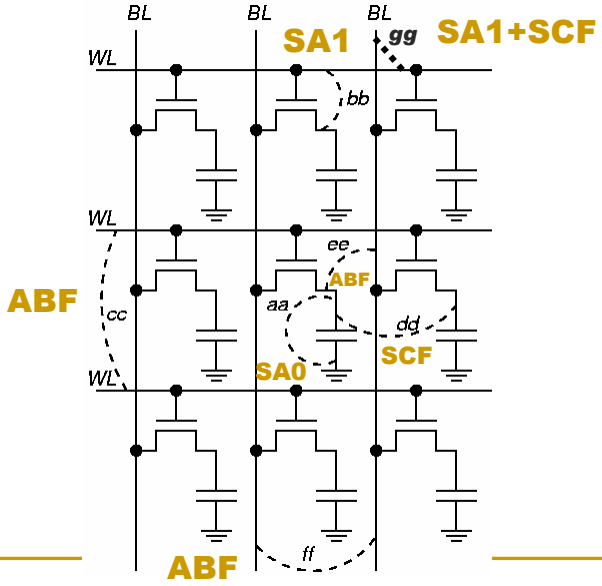


(b) Diagram of a state coupling fault (SCF) <1; 1>.

Fault Modeling Example 1



Fault Modeling Example 2



March Test Notation

- **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**

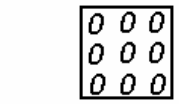
Functional RAM Testing with March Tests

- **March Tests can detect AFs -- NPSF Tests Cannot**
- **Conditions for AF detection:**
  - **Need ⇑(r x, w $\bar{x}$)**
  - **Need ⇓(r $\bar{x}$, w x)**

MATS+ March Test

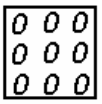
M0: { March element ⇕ (w0) }
for cell := 0 to n - 1 (or any other order) do
write 0 to A [cell];
M1: { March element ⇑ (r0, w1) }
for cell := 0 to n - 1 do
read A [cell]; { Expected value = 0}
write 1 to A [cell];
M2: {March element ⇓ (r1, w0) }
for cell := n – 1 down to 0 do
read A [cell]; { Expected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

MATS+ Example
Cell (2,1) SA0 Fault

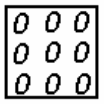


(a) Good machine after M0.

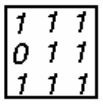
(b) Good machine after M1.



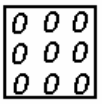
(c) Good machine after M2.



(d) Bad machine after M0.



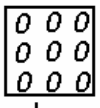
(e) Bad machine after M1.



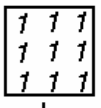
(f) Bad machine after M2.

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

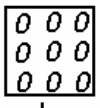
MATS+ Example
Cell (2, 1) SA1 Fault



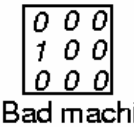
(a) Good machine after M0.



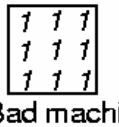
(b) Good machine after M1.



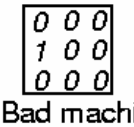
(c) Good machine after M2.



(d) Bad machine after M0.



(e) Bad machine after M1.

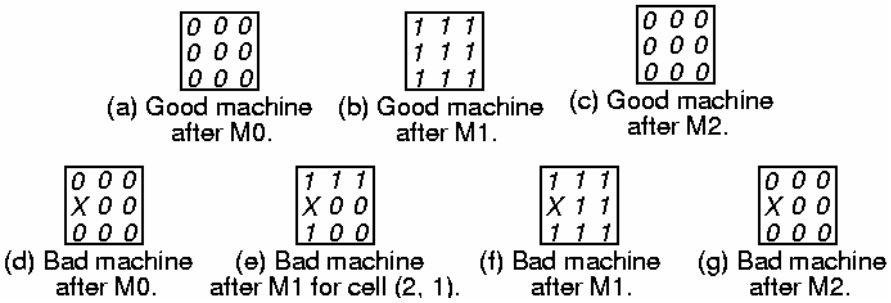


(f) Bad machine after M2.

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

MATS+ Example: Multiple AF 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 #**

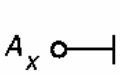
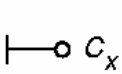
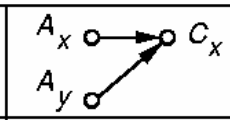


MATS+:

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

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*

Theorem 9.2

- ***A March test satisfying conditions 1 & 2 detects all address decoder faults.***
- **… Means any # of read or write operations**
- **Before condition 1, must have *wx* element**
  - ***x*** **can be 0 or 1, but must be consistent in test**

Condition	March element
1	⇑(rx, …, w $\overline{x}$)
2	⇓(r $\overline{x}$, …, wx)

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) }

Irredundant March Test Summary

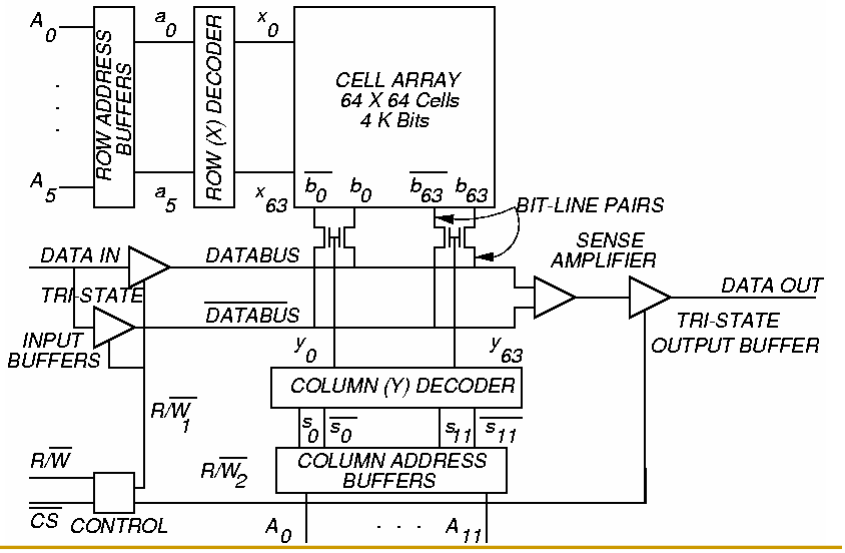
Algorithm	SAF	AF	TF	CF in	CF id	CF dyn	SCF	Linked Faults
MATS	All	Some						
MATS+	All	All						
MATS++	All	All	All					
MARCH X	All	All	All	All				
MARCH C—	All	All	All	All	All	All	All	
MARCH A	All	All	All	All				Some
MARCH Y	All	All	All	All				Some
MARCH B	All	All	All	All				Some

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*

Neighborhood Pattern Sensitive Coupling Faults

RAM Organization

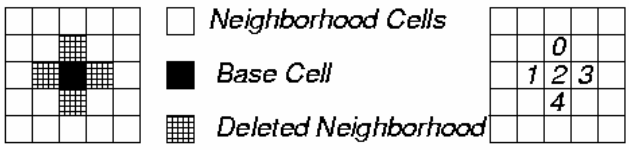


Notation

- **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***

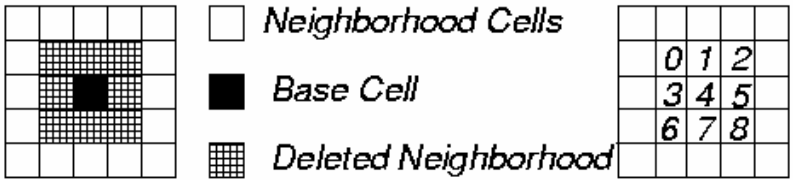
Type 1 Active NPSF

- ***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.**



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.**

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.**

Summary

- Functional and fault model of memory
  - Many fault models
- March tests and their capabilities
  - Variety of tests
- Neighborhood pattern sensitive tests
  - Varity of fault models and tests

Appendix

Density and Defect Trends

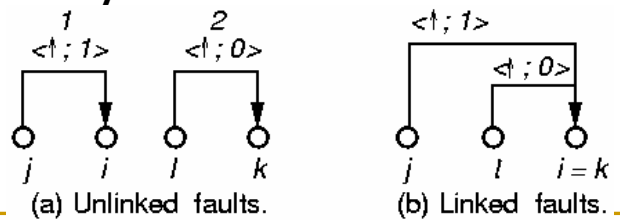
- **1970 -- DRAM Invention (Intel) 1024 bits**
- **1993 -- 1st 256 MBit DRAM papers**
- **1997 -- 1st 256 MBit DRAM samples**
  - **1 ¢/bit --> 120 X 10^{-6} ¢/bit**
- **Kilburn -- Ferranti Atlas computer (Manchester U.) -- Invented Virtual Memory**
  - **1997 -- Cache DRAM -- SRAM cache + DRAM now on 1 chip**

Physical Failure Mechanisms

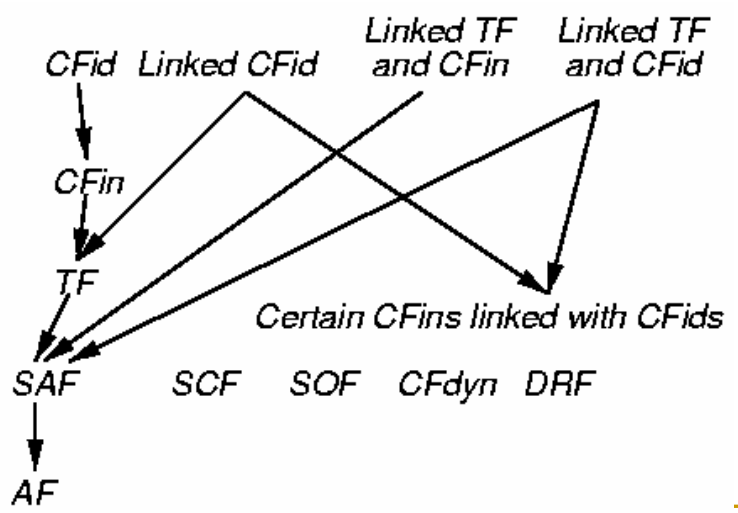
- **Corrosion**
- **Electromigration**
- **Bonding Deterioration -- *Au* package wires interdiffuse with *Al* chip pads**
- **Ionic Contamination -- Na^+ diffuses through package and into FET gate oxide**
- **Alloying -- *Al* migrates from metal layers into *Si* substrate**
- **Radiation and Cosmic Rays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

Multiple Fault Models

- **Coupling Faults: In real manufacturing, any # can occur simultaneously**
- **Linkage: A fault influences behavior of another**
- **Example March test that fails:**
  - **{⇕(w0) ; ⇑(r0, w1); ⇕(w0, w1); ⇕(r1)}**
  - **Works only when faults not linked**

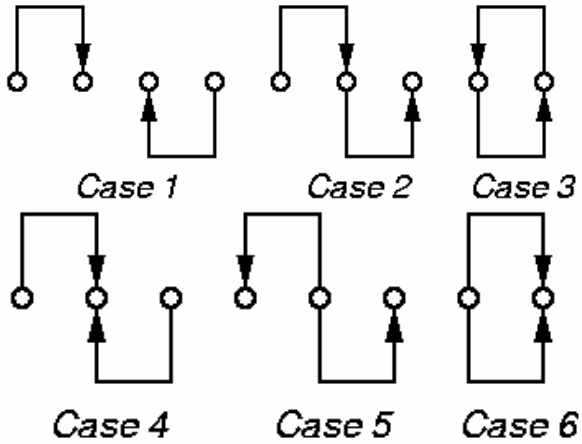


Fault Hierarchy



Tests for Linked AFs

- **Cases 1, 2, 3 & 5 -- Unlinked**
- **Cases 4 & 6 -- Linked**



DRAM/SRAM Fault Modeling

DRAM or SRAM Faults	Model
Shorts & opens in memory cell array	SAF,SCF
Shorts & opens in address decoder	AF
Access time failures in address decoder	Functional
Coupling capacitances between cells	CF
Bit line shorted to word line	IDDQ
Transistor gate shorted to channel	IDDQ
Transistor stuck-open fault	SOF
Pattern sensitive fault Diode-connected transistor 2 cell short Open transistor drain Gate oxide short Bridging fault	PSF

SRAM Only Fault Modeling

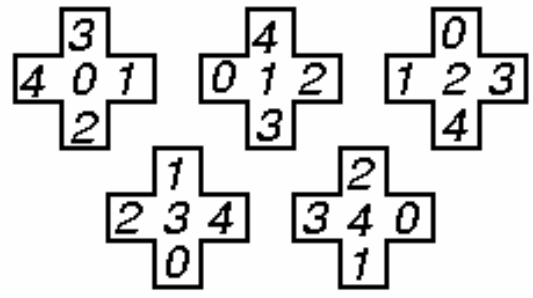
Faults found only in SRAM	Model
Open-circuited pull-up device	DRF
Excessive bit line coupling capacitance	CF

DRAM Only Fault Modeling

Faults only in DRAM	Model
Data retention fault (sleeping sickness)	DRF
Refresh line stuck-at fault	SAF
Bit-line voltage imbalance fault	PSF
Coupling between word and bit line	CF
Single-ended bit-line voltage shift	PSF
Precharge and decoder clock overlap	AF

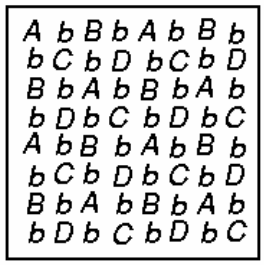
Type 1 Tiling Neighborhoods

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

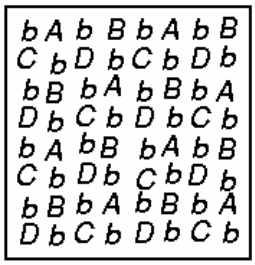


Two Group Method

- **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.

RAM Tests for Layout-Related Faults

***Inductive Fault Analysis*:**

1. **Generate defect sizes, location, layers based on fabrication line model**
2. **Place defects on layout model**
3. **Extract defective cell schematic & electrical parameters**
4. **Evaluate cell testing**

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**
- ***Inductive Fault Analysis* is now required**