



# VLSI Testing

## 積體電路測試

### *Memory Testing*

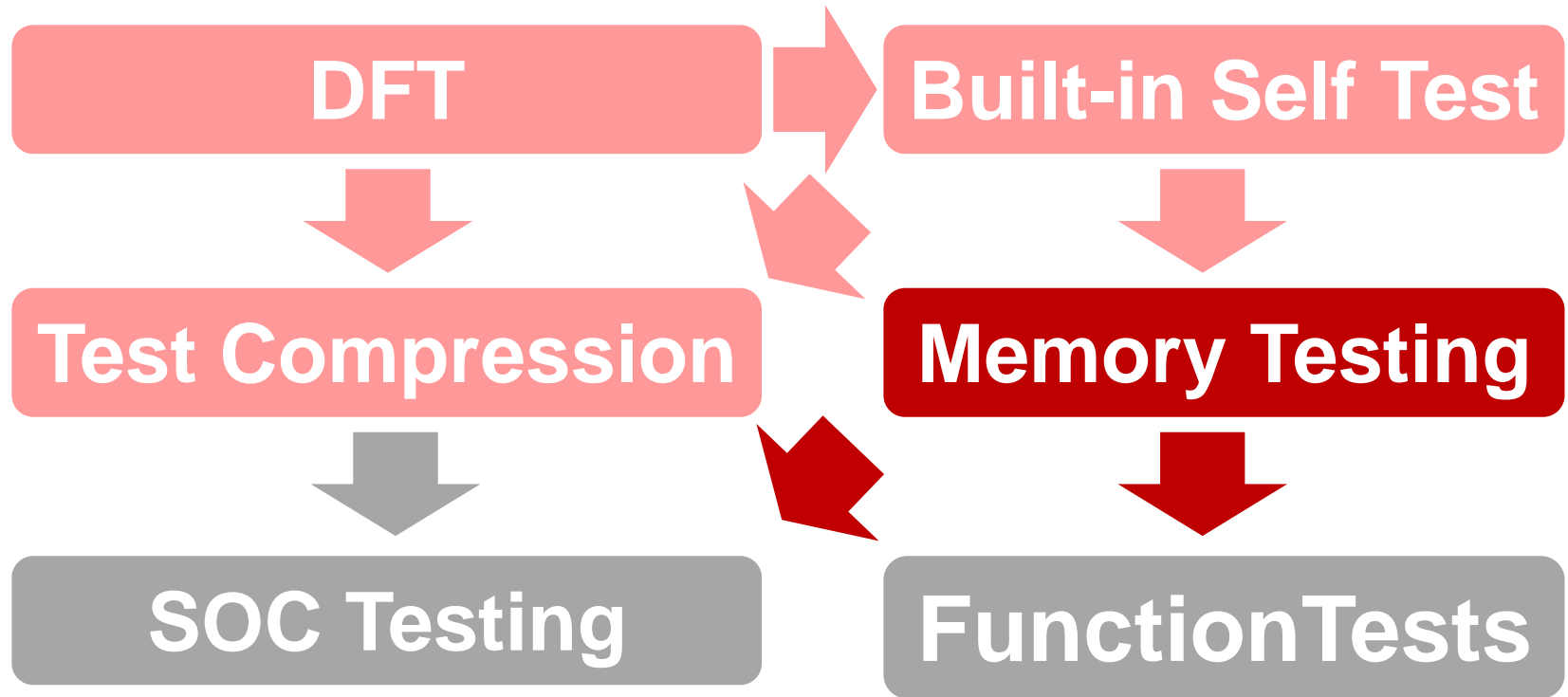
**Professor James Chien-Mo Li 李建模**

**Lab. of Dependable Systems**

**Graduate Institute of Electronics Engineering**

**National Taiwan University**

# Course Roadmap (Design Topics)



# Why Am I Learning This?

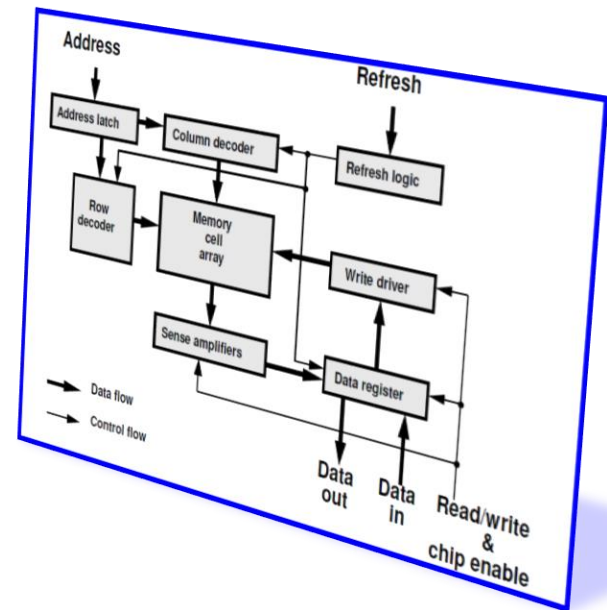
- Testing memory is important
  - ♦ Because many memories are needed in modern designs
- Memory testing is very different from logic testing

*“The advantage of a bad memory is that one enjoys several times the same good things for the first time”*

*( Friedrich Nietzsche )*

# Outline

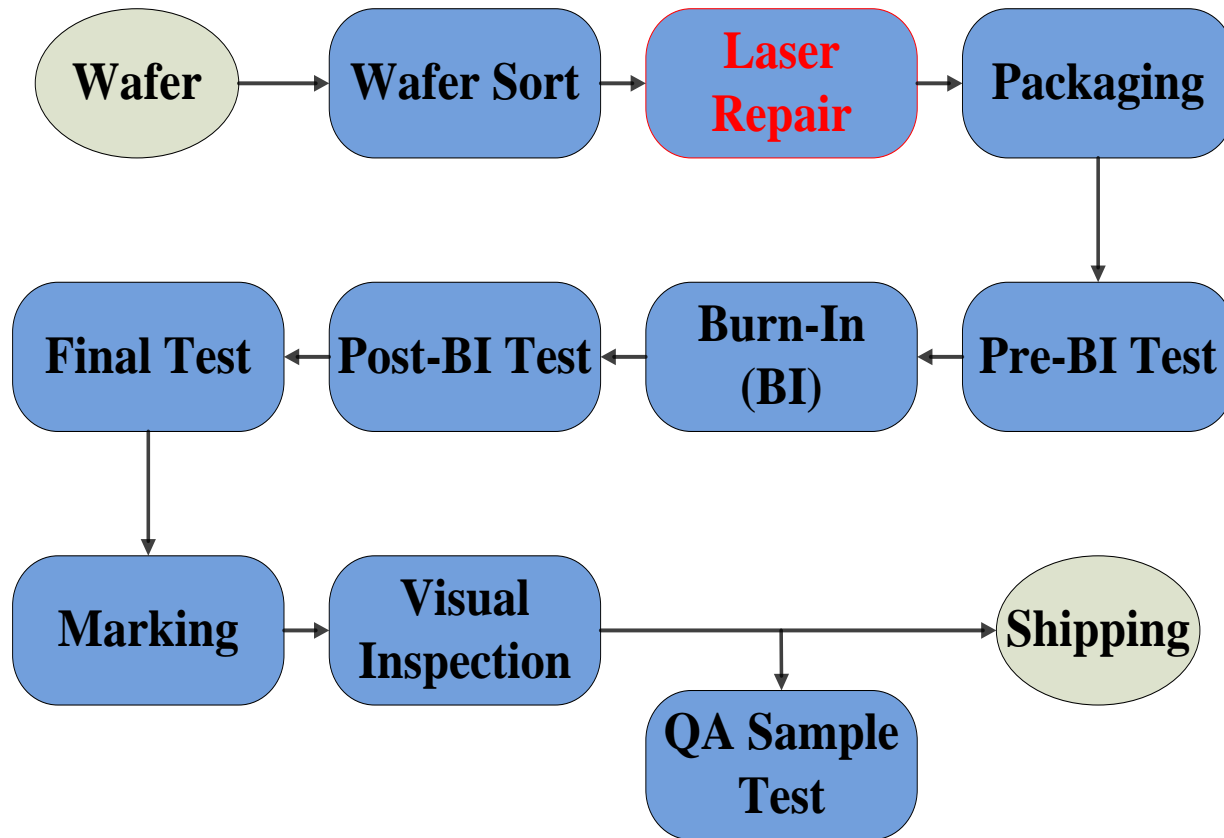
- Introduction
- Memory Fault Models
- Memory Test Algorithms
- Memory Fault Simulation (\*not in exam)
- Memory Test Generation (\*not in exam)
- Memory BIST (\*not in exam)



# Memory Testing

- Testing of memory is crucial for quality. Research started 1960's
  - ◆ Memory accounts for ~30% of semiconductor market (2019)
- Popular memory test items
  - ◆ Off-chip tests
    - \* DC parametric test: e.g. leakage current, output voltage level
    - \* AC parametric test: e.g. rise time, fall time
    - \* Functional test: e.g. march test (see 16.3)
    - \* Retention test: measure retention time of DRAM
    - \* Reliability test (Burn-in)
  - ◆ On-chip tests
    - \* *Error Detection and Correction (EDAC)*: on-line testing
    - \* Built-in Self Test (BIST)
    - \* *Built-in Self Diagnosis & Repair (BISDR)*
- BIST important for System-on-chip (SOC) with *embedded memories*

# Typical Memory Test Flow

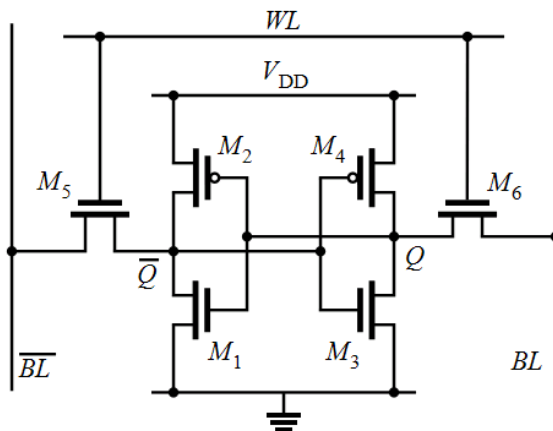


**Memory Diagnosis/Repair can Improve Yield**

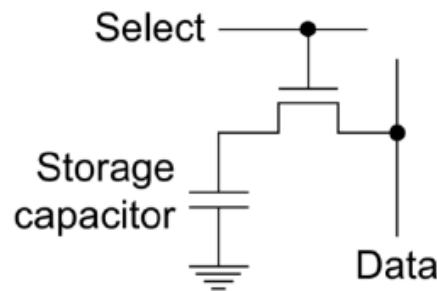
# Types of Memories



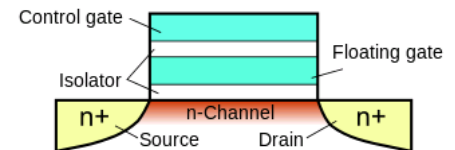
type	area	speed	retention	application	test method
<b>SRAM</b>	largest 6T	<b>fastest</b> <1ns	as long as power	embedded SRAM cache, registers	BIST
<b>DRAM</b>	medium 1T+1C	medium ~10ns	< sec.	embedded DRAM	BIST/ ATE
				on-board memory	ATE
<b>Flash</b>	<b>smallest</b> 1T	<b>slowest</b> ~100μs	years	SSD, USB drive	ATE



SRAM cell [wikipedia]

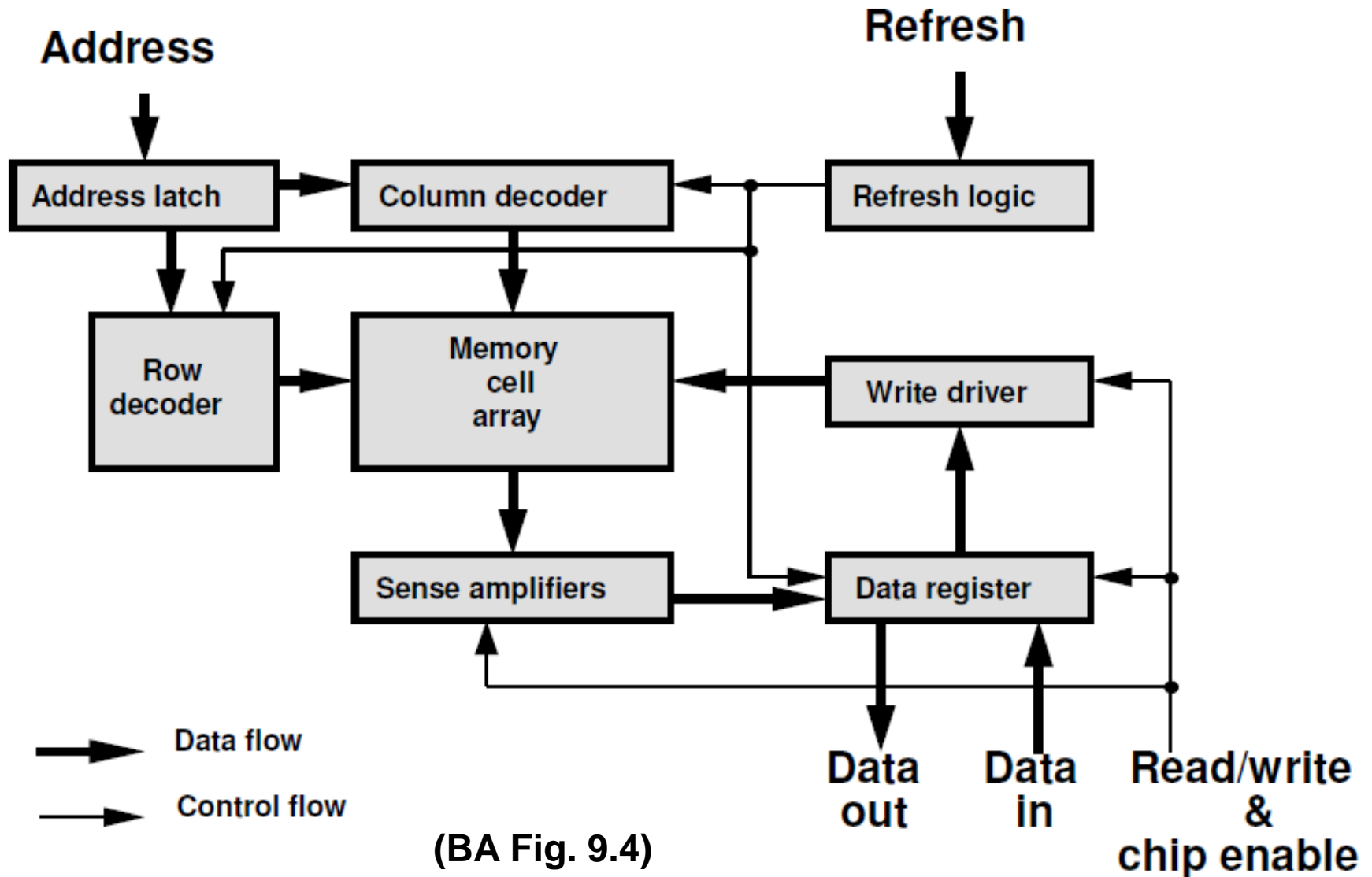


DRAM cell



Floating Gate Transistor [wikipedia]

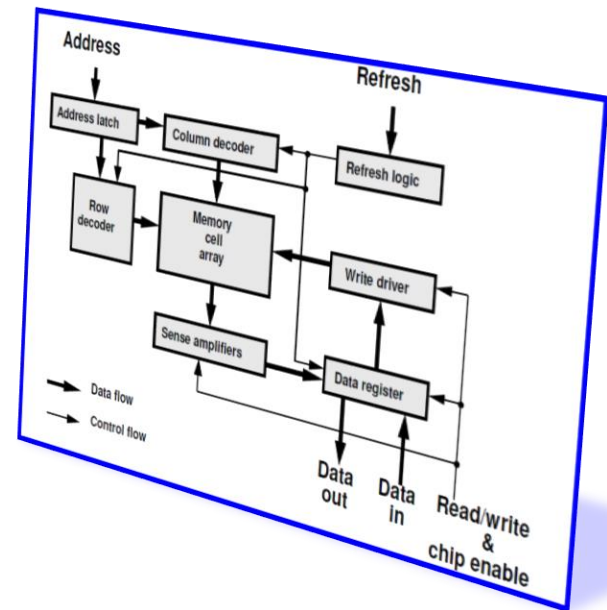
# DRAM Functional Model





# Outline

- Introduction
- Memory Fault Models (focus on RAM)
  - ◆ Static faults
    - \* Single cell fault, Double cell fault, Address-decoder fault
  - ◆ Dynamic faults
    - \* Recovery fault, Retention fault
- Memory Test Algorithms
- Memory Fault Simulation (\*not in exam)
- Memory Test Generation (\*not in exam)
- Memory BIST (\*not in exam)



# RAM Fault Models

- **Functional testing** is commonly used for memories
  - ♦ Scan testing for logic is not applicable to memory. (Why? FFT)
- **Functional fault models** are behavior model for faulty memories
  - ♦ based on **real defects**, not imagination
- Popular **RAM functional fault models**
  1. **Static** fault models: faulty behavior does NOT change with time
    - \* single cell, double cell, address decoder
  2. **Dynamic** fault models: faulty behavior changes with time
    - \* recovery, data retention
- NOTE: many other RAM fault models (neighborhood pattern sensitive faults ...)

**Different Memories Need Different Fault Models**

# RAM Fault Models (1) – single cell

- **Stuck-At Fault (SAF)**
  - ♦ a cell is always 0, **SA0**
  - ♦ a cell is always 1, **SA1**
- **Stuck-Open Fault (SOF)**
  - ♦ a cell cannot be accessed due to broken wire
- **Transition Fault (TF)**
  - ♦ a cell fails to
    - \* Rise from 0 to 1     **<↑ / 0>**
    - \* Fall from 1 to 0     **<↓ / 1>**

NOTATION: **<S/F>**: a fault in a cell [van de Goor 91]

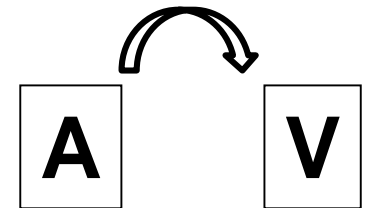
- ♦ **S** is value or operation activating fault ,      $S \in \{0, 1, \uparrow, \downarrow, \updownarrow, \forall\}$ 
  - \* **↑** is rising; **↓** is falling; **↑↓** is either **↑** or **↓**; **∀** means any condition
- ♦ **F** is faulty value of cell ,      $F \in \{0, 1, \updownarrow\}$ 
  - \* **↑↓** is complement

# RAM Fault Models (2) – double cell

- **Coupling Fault (CF):** *Victim* cell is affected by *Aggressor* cell
  - ♦ **1.State Coupling Fault ( $CF_{st}$ )**
    - \* if aggressor cell is in given state, victim cell is forced to 0 or 1
    - \* 4 types:  $\langle 0; 0/1 \rangle$  or  $\langle 0; 1/0 \rangle$  or  $\langle 1; 1/0 \rangle$  or  $\langle 1; 0/1 \rangle$
  - ♦ **2.Inversion Coupling Fault ( $CF_{in}$ )**
    - \* if aggressor cell rise/fall, victim cell is complemented
    - \* 2 types:  $\langle \uparrow; \nabla/\downarrow \rangle$  or  $\langle \downarrow; \nabla/\uparrow \rangle$
  - ♦ **3.Idempotent Coupling Fault ( $CF_{id}$ )**
    - \* if aggressor cell rise/fall, victim cell is forced to 0 or 1
    - \* 4 types:  $\langle \uparrow; 0/1 \rangle$  or  $\langle \uparrow; 1/0 \rangle$  or  $\langle \downarrow; 0/1 \rangle$  or  $\langle \downarrow; 1/0 \rangle$

NOTATION:  $\langle S_1; S_2/F \rangle$  faults in 2 cells

- ♦  $S_1$  is value or operation activating fault in aggressor
- ♦  $S_2$  is value or operation activating fault in victim
- ♦  $F$  is faulty value of victim



# CF Examples

$CF_{st} < 0, 0/1 >$

addr	content*
A	0
V	0

↓ do nothing

addr	content
A	0
V	0/1

↓ read V

detected

\*typically many bits in one address  
but only one bit here for illustration

$CF_{id} < \uparrow; 0/1 >$

A	0
V	0

↓ write 1 to A

A	1
V	0/1

↓ read V

detected

A	0
V	1

↓ write 1 to A

A	1
V	1

↓ read V

not detected

$CF_{in} < \uparrow; \forall \updownarrow >$

A	0
V	0

↓ write 1 to A

A	1
V	0/1

↓ read V

detected

A	0
V	1

↓ write 1 to A

A	1
V	1/0

↓ read V

detected

# QUIZ


Q: Consider  $CF_{in} \langle \downarrow, \uparrow \rangle$ . Fill in values for two cases.  
Can we detect faults in both cases?

ANS:


ad dr	content
A	1
V	1

 write 0  
to A

ad dr	content
A	0
V	?

 read  
V=?

ad dr	content
V	1
A	1

 write 0  
to V

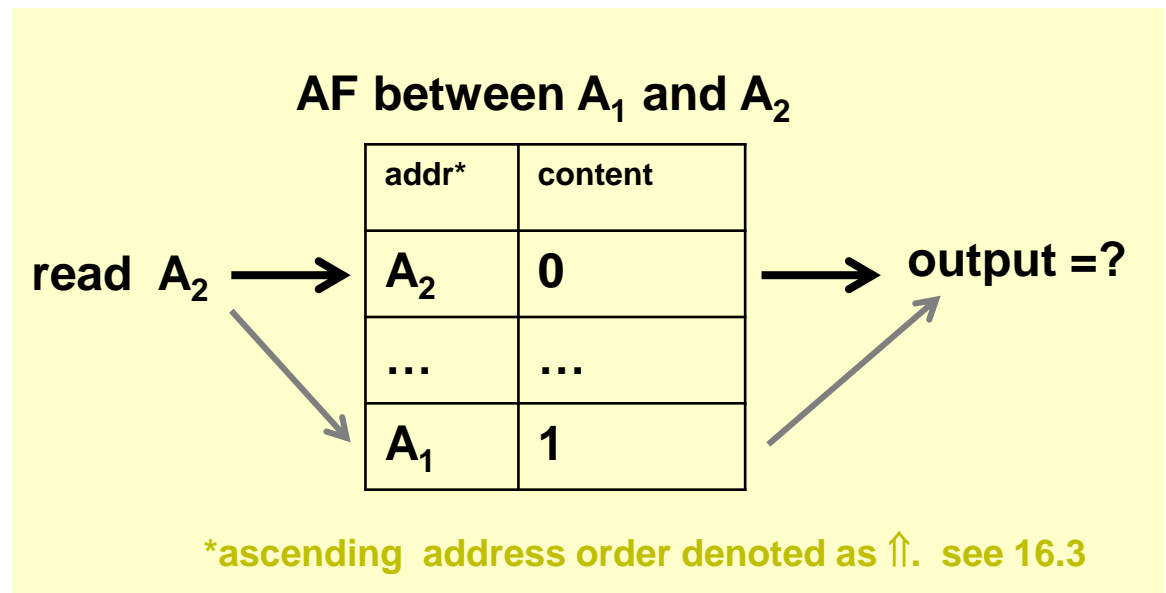
ad dr	content
V	0
A	?

 read  
A=?

**Address Order Matters**

# RAM Fault Models (3) – AF

- **Address-Decoder Fault (AF)** Four faulty behavior:
  1. Given a certain address, no cell will be accessed
  2. A certain cell is never accessed by any address
  3. A certain cell can be accessed by multiple addresses
  - ★ 4. Given a certain address, multiple cells are accessed
    - **AND-type**
    - **OR-type**



**Only Consider #4 AF**

# AF Examples

## OR-type AF

between  $A_1$  and  $A_2$

$A_2$	1
...	...
$A_1$	0

⇒ read  
 $A_2 = 1$

⇒ read  
 $A_1 = 0/1$

## AND-type AF

between  $A_1$  and  $A_2$

$A_2$	1
...	...
$A_1$	0

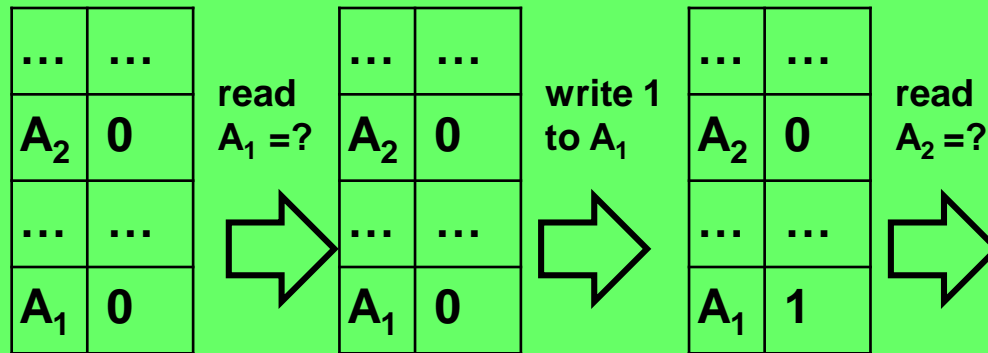
⇒ read  
 $A_2 = 1/0$

⇒ read  
 $A_1 = 0$

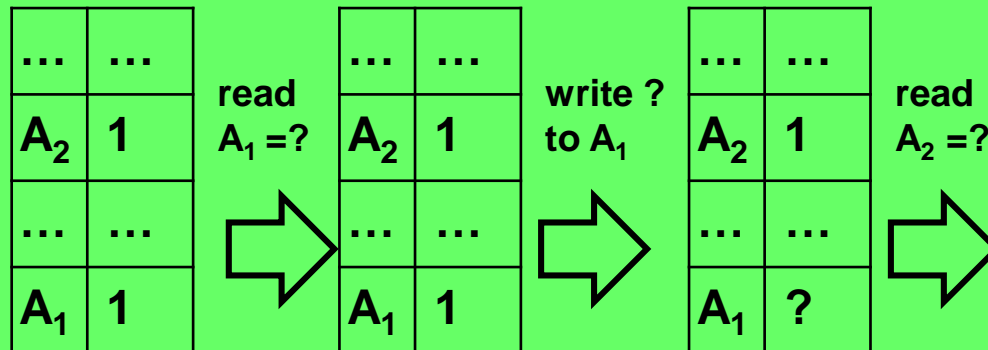


# QUIZ

Q1: Given **OR-type** AF between  $A_1$  and  $A_2$ . Fault detected?



Q2: Given **AND-type** AF. Find a test to detect fault.



**AND/OR AF Need Opposite Test Data**

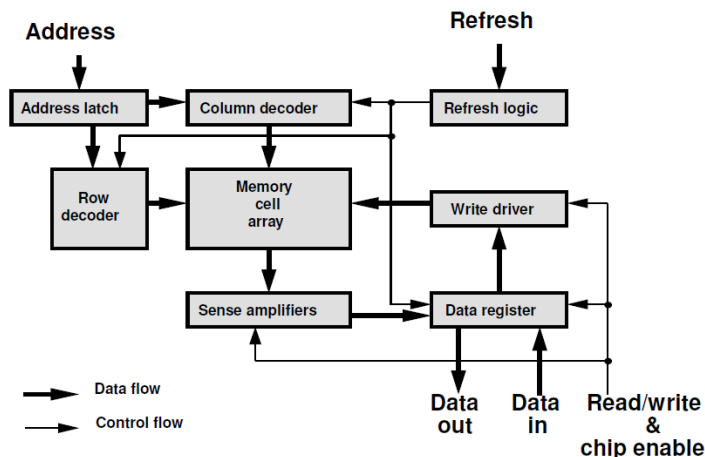
# RAM Fault Models (4) – dynamic faults

- ***Data Retention Fault (DRF):*** data changed after a certain time
  - ♦ **DRAM**
    1. Charge leakage loose data in capacitor
    2. Refresh logic fails to refresh correctly
  - ♦ **SRAM**
    - \* Defective pull-up device inducing excessive leakage current which changes the state of cell
- ***Sense amplifier recovery fault***
  - ♦ Sense amp. saturated after reading/writing a long string of 0 or 1
- ***Write recovery fault***
  - ♦ A write followed by a read/write at a different location results in reading or writing at the same location due to slow address decoder

**Time Consuming to Test Dynamic Faults**

# Summary

- Memory test important. Good diagnosis/repair can **improve yield**
- Popular **RAM functional fault models**
  - ◆ **Static** fault models
    - \* Single cell: Stuck-at (**SAF**), Stuck-open (**SOF**), Transition (**TF**)
    - \* Double cell coupling faults: **CF<sub>in</sub>**, **CF<sub>id</sub>**, **CF<sub>st</sub>**
    - \* Address decoder fault (**AF**): **AND**-type **OR**-type
  - ◆ **Dynamic** fault models
    - \* Data retention faults (**DRF**). Recovery faults
- Fault model must be **realistic**
  - ◆ Different memories need **different** fault models



# FFT

- Q1: Scan testing for logic is not applicable to memory.
  - ◆ Why no scan?
- Q2: AF has four faulty behavior.
  - ◆ We only consider #4. The others are easy to test, why?
    1. Given a certain address, no cell will be accessed
    2. A certain cell is never accessed by any address
    3. A certain cell can be accessed by multiple addresses
    - ★4. Given a certain address, multiple cells are accessed