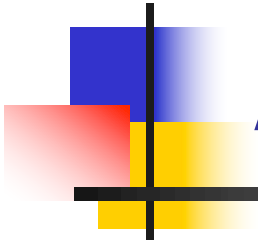
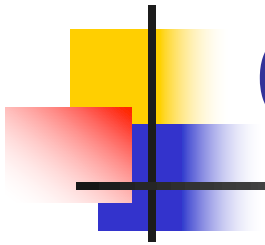


Chapter 10
Memory Testing

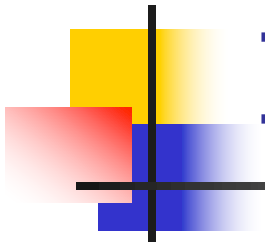
Arnaud Virazel

virazel@lirmm.fr



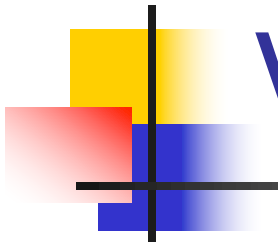
Outline

- **Introduction**
- Memory modelling
- Failure mechanisms and Fault Modelling
- Test algorithms
- Memory BIST



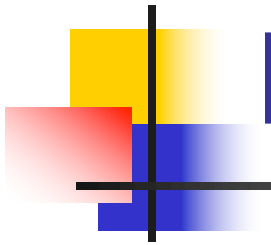
Introduction

- Memories (especially SRAM) are at the forefront of commercial circuit design
- DRAMs are driving technological development in the semiconductor industry
- Memories are the most used cores in SoCs



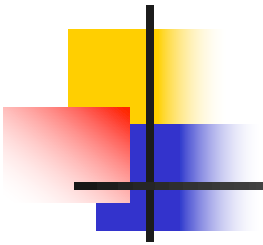
Volitile Memory Types

- Dynamic RAM (DRAM)
  - Highest density
  - Low speed access time (about 10ns)
  - Information is stored as the charge of a capacitor and must be refreshed regularly
- Static RAM (SRAM)
  - The fastest (about 1ns)
  - The information is stored in latches made with looped inverters



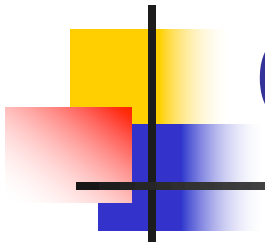
Non-Volatile Memory Types

- Read Only Memory (ROM)
  - Information is stored by the presence or absence of a transistor during manufacture
  - The information persists even if the circuit is not powered
- Erasable and Programmable Read Only Memory (EPROMs)
  - Programmable in the application
  - Fully erasable by applying ultraviolet rays
- Electrically Erasable and Programmable RO (EEPROM, FLASH)
  - Data can be selectively erased by electrical means



Memory Test

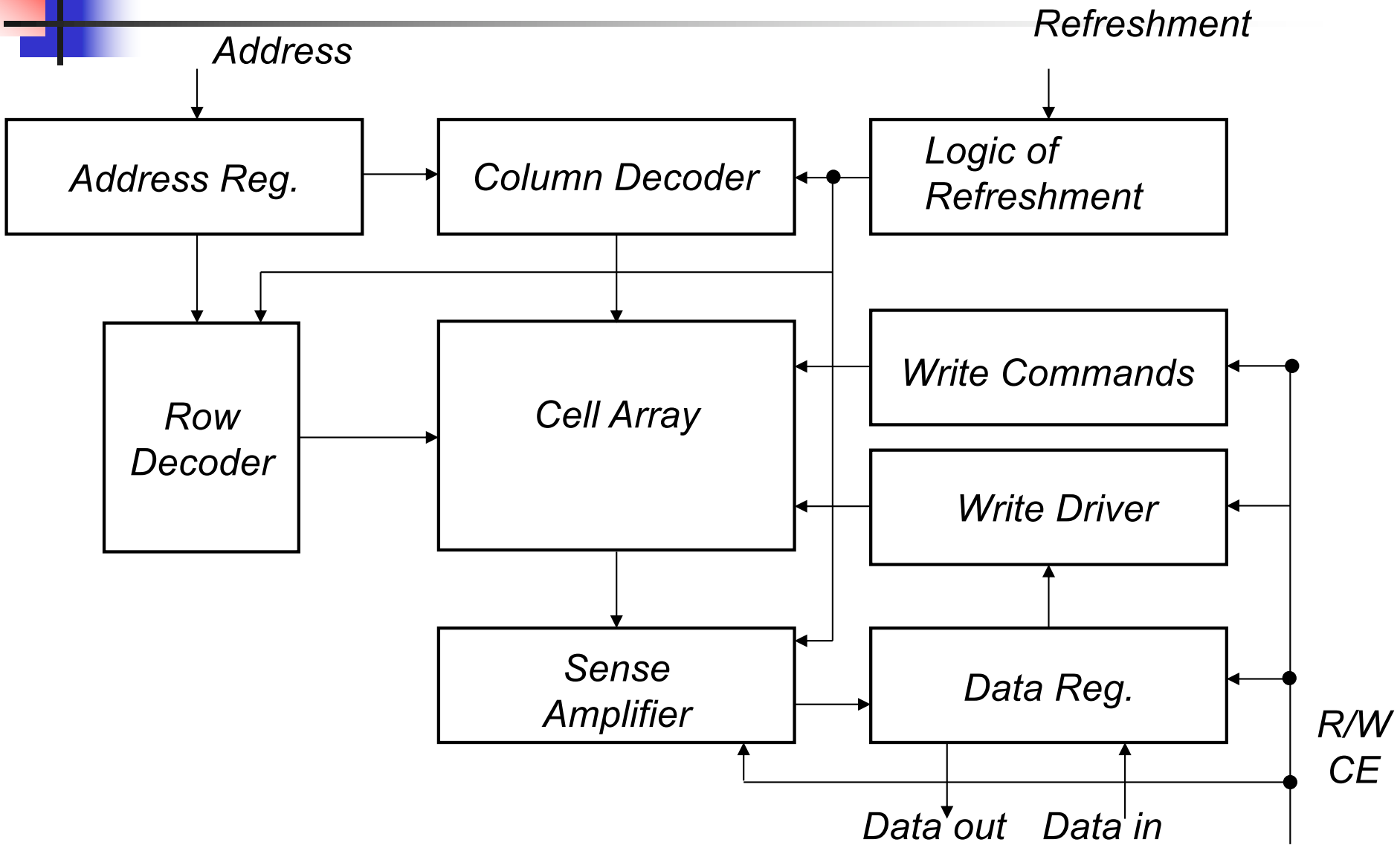
- Memory test must prove that the circuit under test behaves as it was designed, so it consists of:
  - parametric tests relating to the level of currents/voltages and delays on the I/O pins of the circuit
  - a functional test
    - Modelling of functional faults



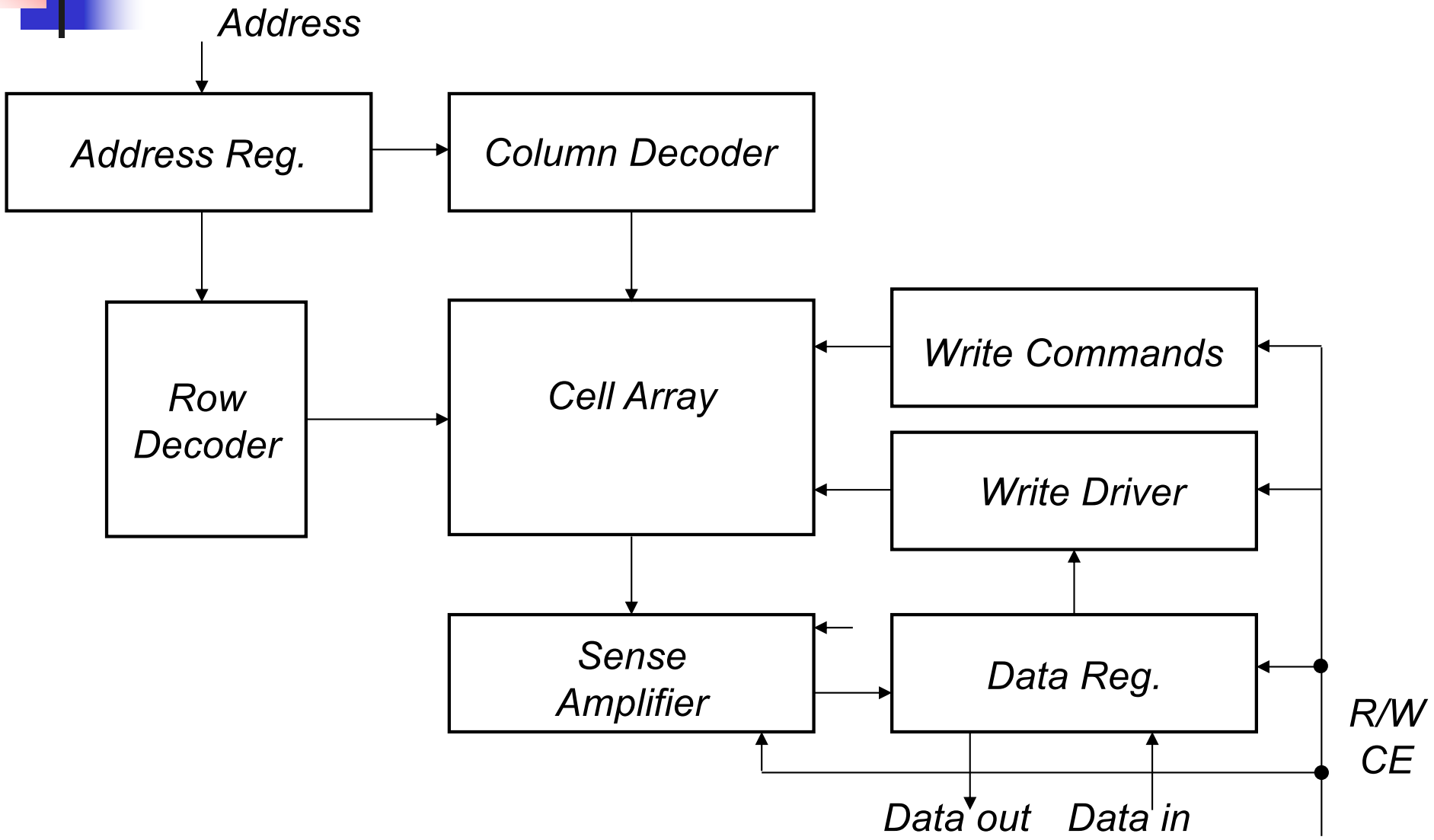
Outline

- Introduction
- **Memory modelling**
- Failure mechanisms and Fault Modelling
- Test algorithms
- Memory BIST

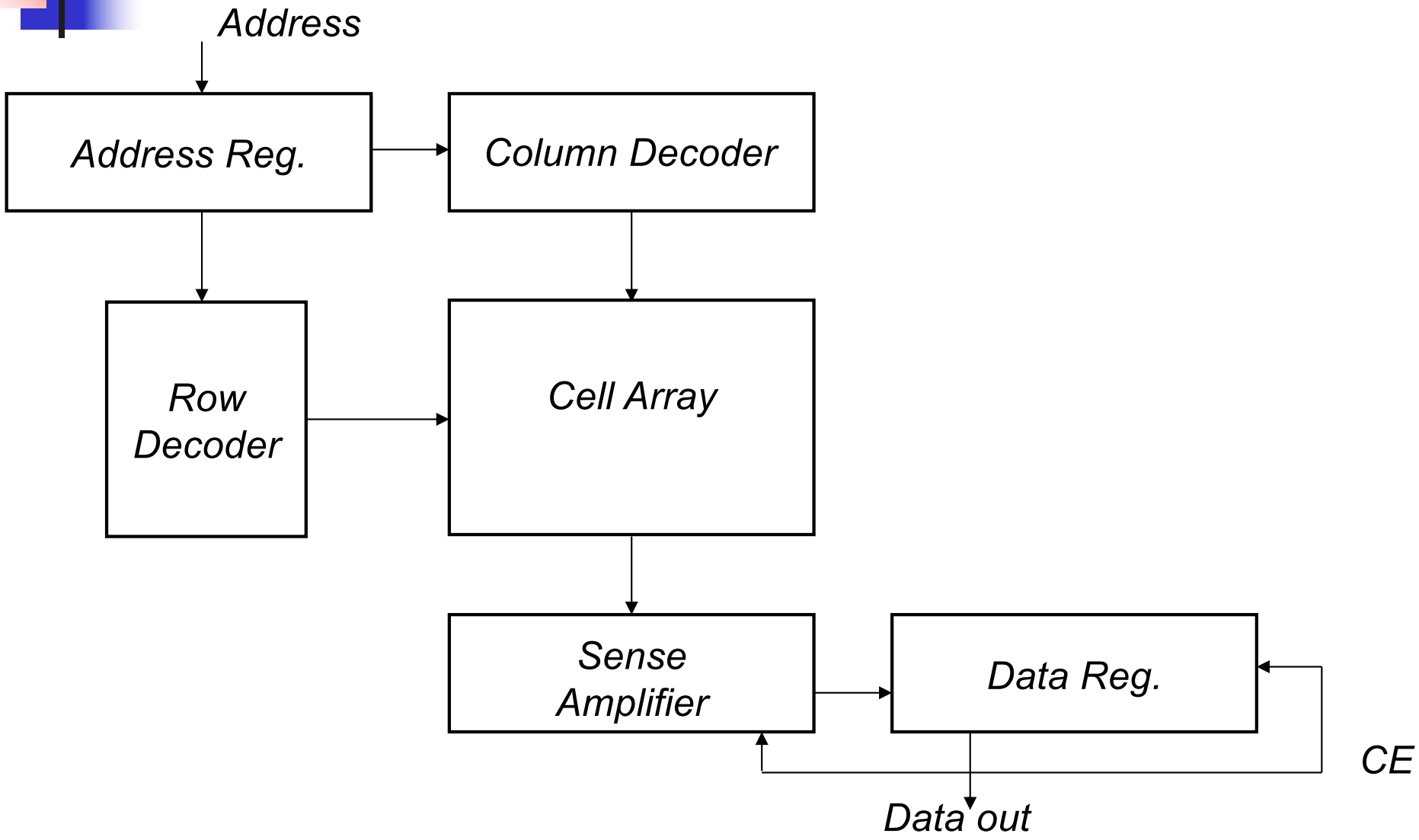
Functional Model - DRAM

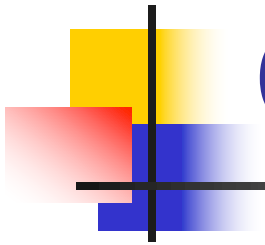


Functional Model - SRAM



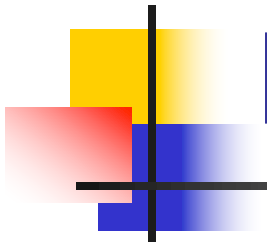
Functional Model - ROM





Outline

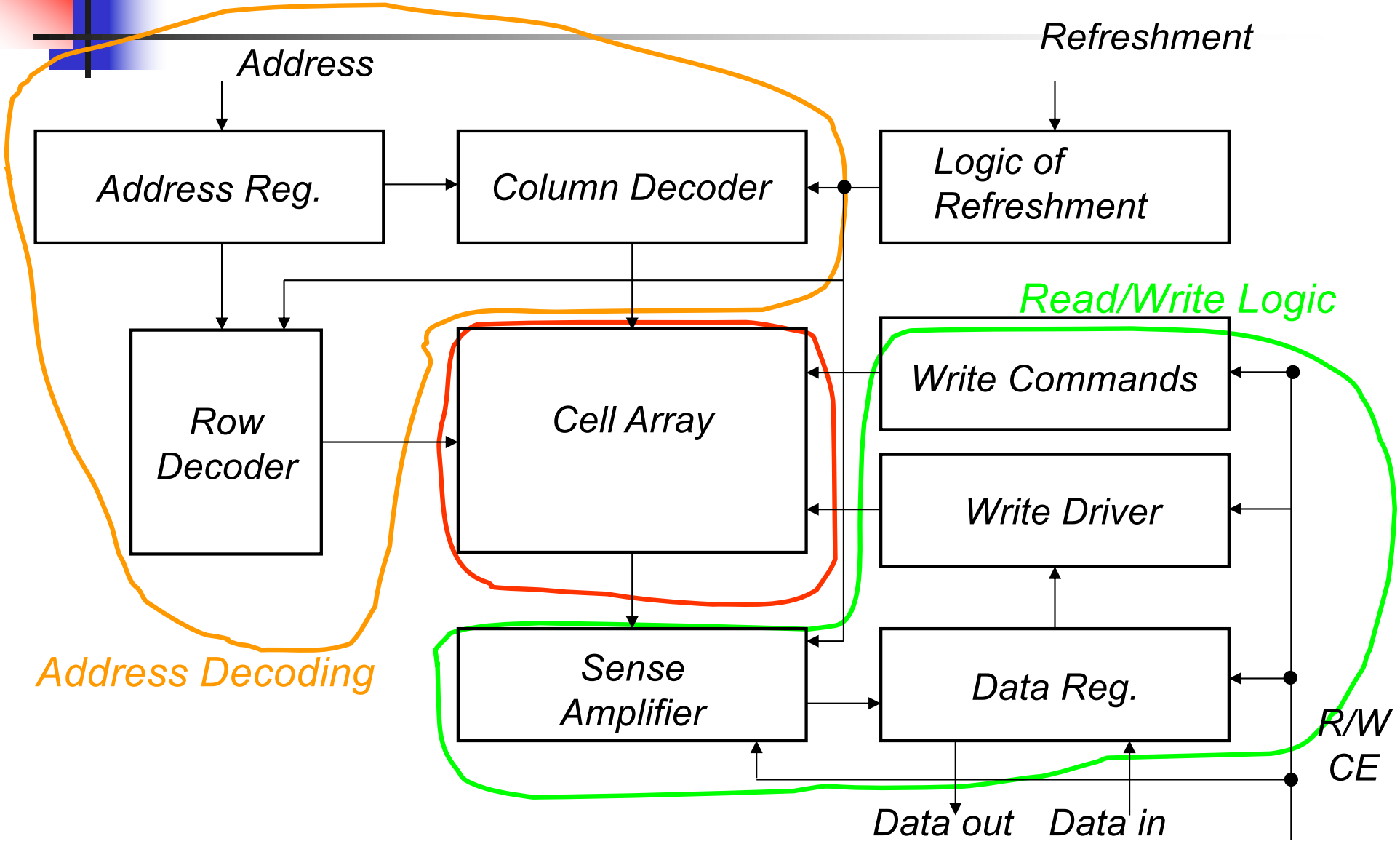
- Introduction
- Memory modelling
- **Failure mechanisms and Fault Modelling**
- Test algorithms
- Memory BIST



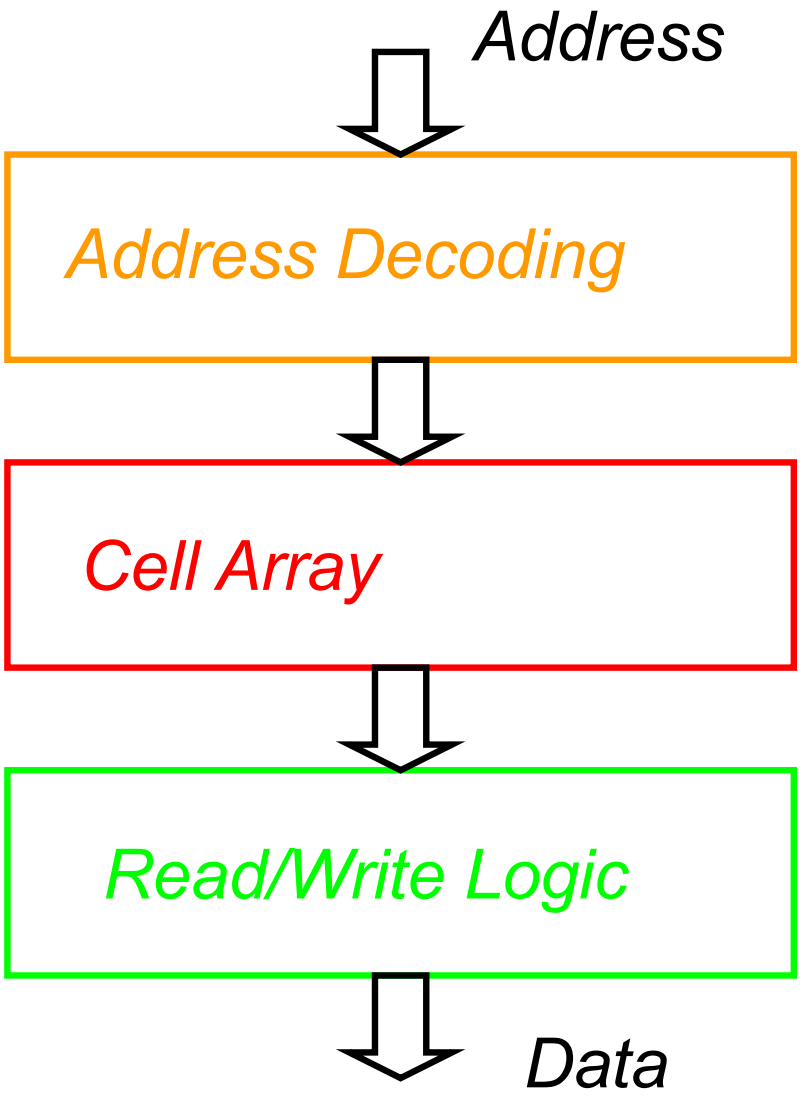
Functional Faults

- Stuck of cell
- Stuck of diver
- Stuck of line
- Stuck of selection line
- Stuck of data line
- Open of data line
- Short of data lines
- Crosstalk of data line
- Stuck of address line
- Open of address ligne
- Short of address line
- Open of decoder
- Bad access
- Multiple access
- A cell can be set to 0 and not to 1
- Interaction between cells

Reduce Functional Model



Reduce Functional Model



Reduce Funtional Faults

- SAF → Stuck-At Fault
- TF → Transition Fault

} *Single cell*

- CF → Coupling Fault

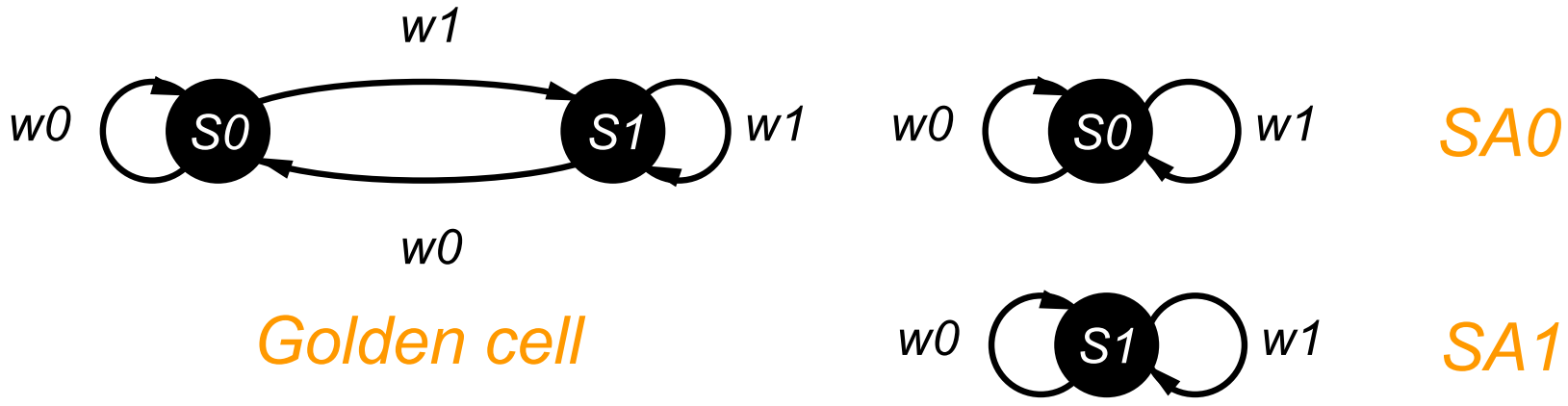
} *2 or more cells*

- NPSF → Neighbourhood Faults

} *K cells*

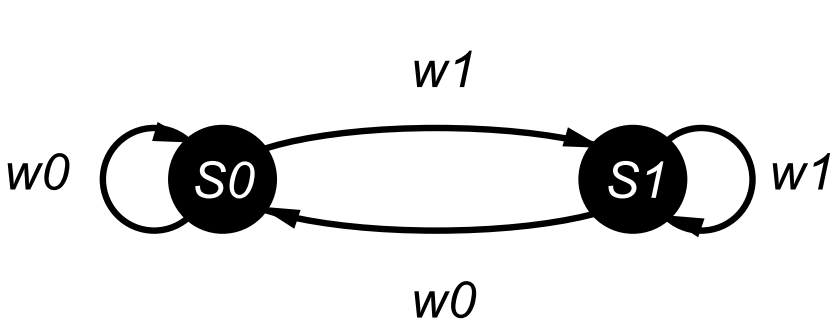
Stuck-at-Fault

- The logic contain of a cell is always stuck-at 0 (SA0) or at 1 (SA1)
- To detect SAF of a memory cell
  - SA0 : Write 1 Read 1 (w1 r1)
  - SA1 : Write 0 Read 0 (w0 r0)

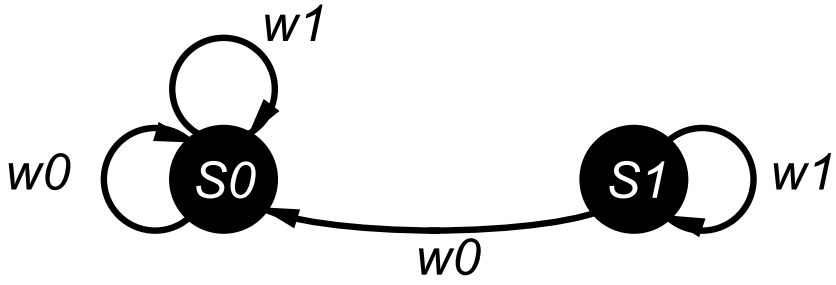


Transition Fault

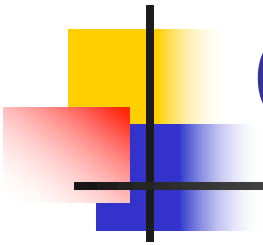
- A memory cell cannot undergoes
  a transition 0 $\rightarrow$ 1 (TFrise) or
  a transition 1 $\rightarrow$ 0 (TFfall)
- To detect TF of a memory cell
  - TFrise : w0 w1 r1
  - TFfall : w1 w0 r0



Golden cell



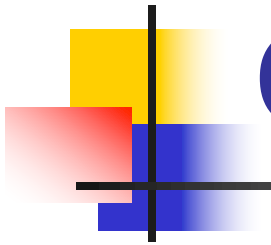
Cell affected by a TF_{rise}



Coupling Faults (2-cell)

- Imply two cells: a victim-cell and an aggressor-cell
- Possible coupling fault models:
  - Inversion
  - Idempotent
  - State coupling
  - Short-circuit
  - Dynamic coupling
  - …

CF Inversion

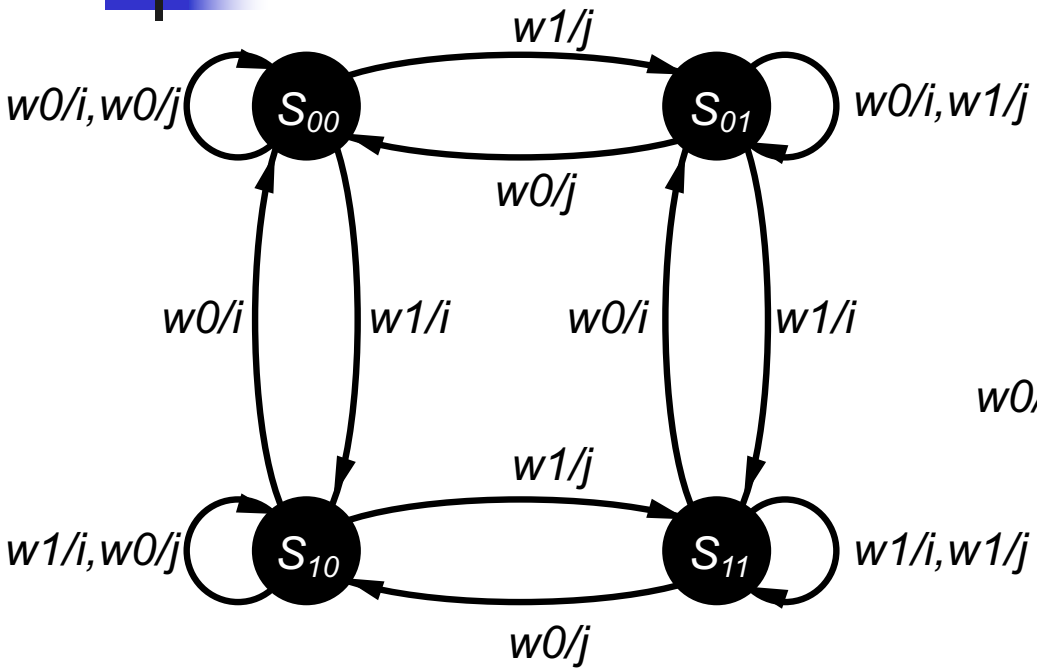


- CFin: The content of the victim cell is inverted by a transition from the aggressor cell
- Depending on the transition ($0 \rightarrow 1$ or $1 \rightarrow 0$), there is two types of CFin:

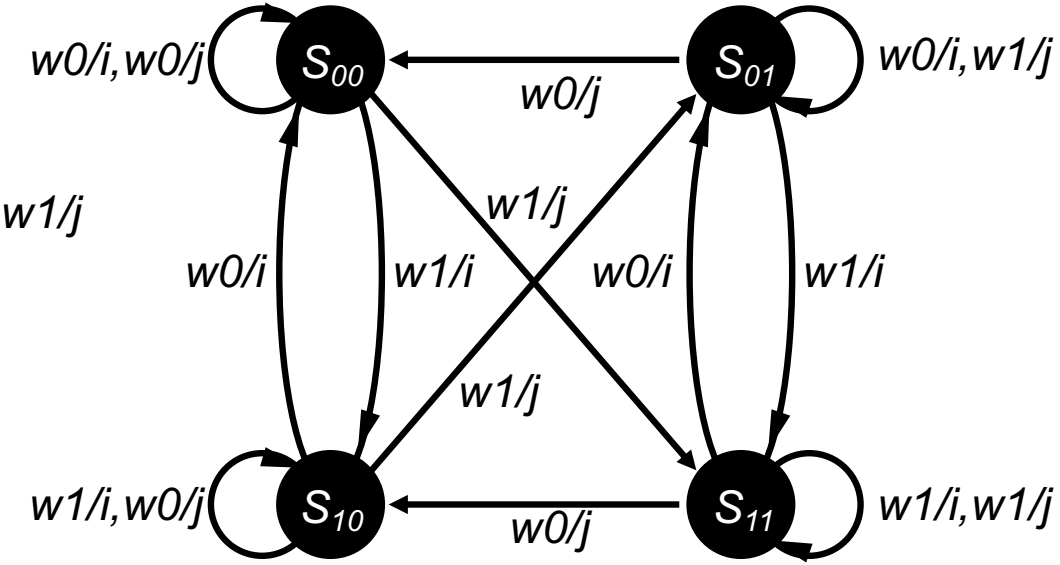
  $< \uparrow ; \updownarrow > < \downarrow ; \updownarrow >$

- To detect CFin between a cell x (victim) and a cell y (aggressor)
  - CFin (y rise $\rightarrow$ x inverted) : w0x w0y w1y r0x.
  - CFin (y fall $\rightarrow$ x inverted) : w0x w1y w0y r0x.

CF Inversion



Two golden cells



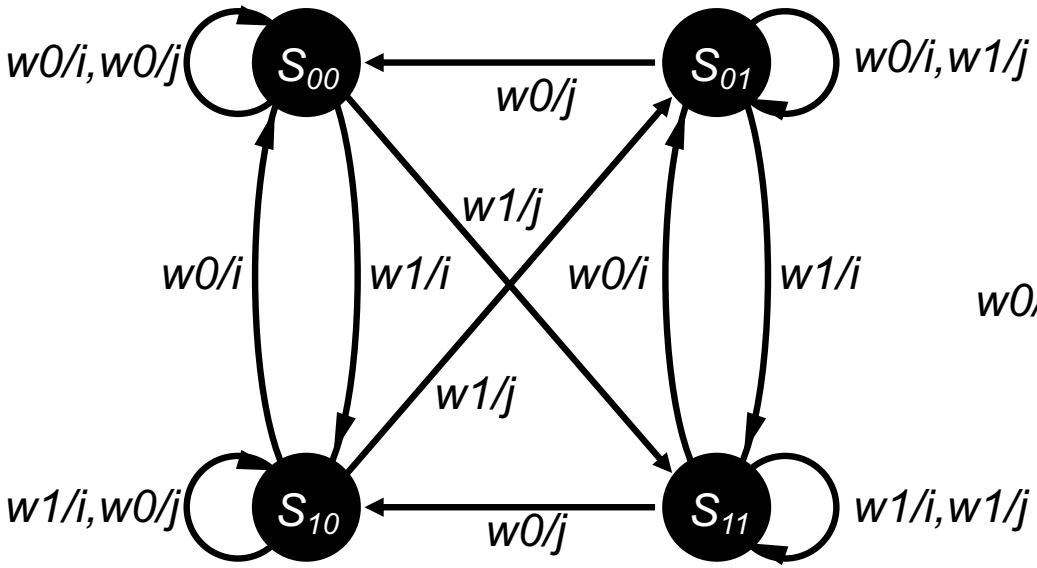
Two cells affected by a <↑;↕> CF_{in}

CF Idempotent

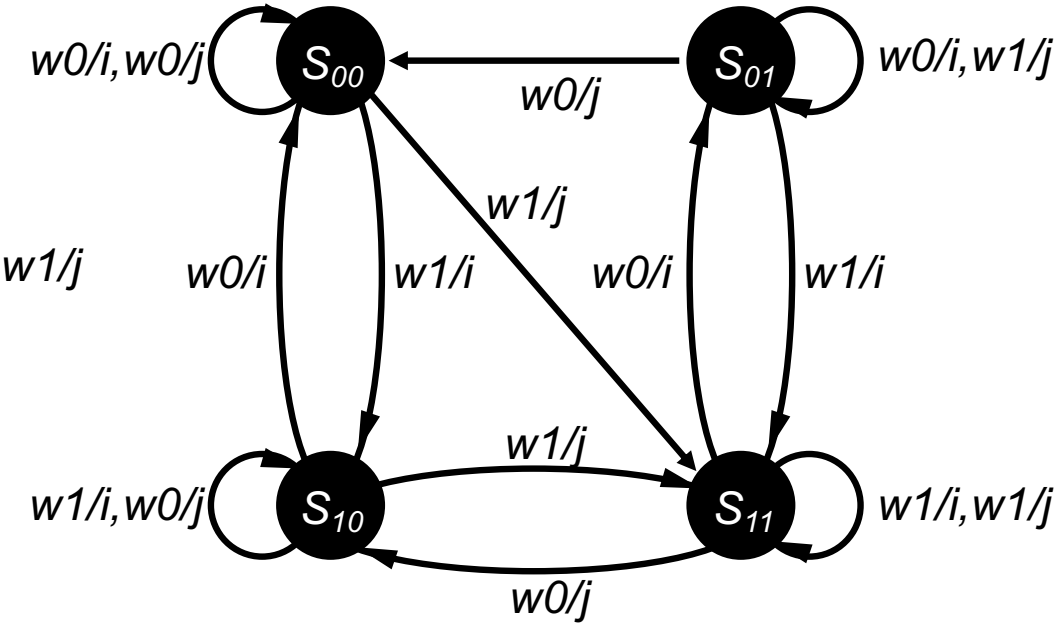
- CFid: The victim cell is forced to 0 or 1 is the aggressor-cell undergoes a transition $0 \rightarrow 1$ or $1 \rightarrow 0$
- Depending on the transition ($0 \rightarrow 1$ or $1 \rightarrow 0$), there is - types of CFid:

  $< \uparrow ; 0 > < \downarrow ; 0 > < \uparrow ; 1 > < \downarrow ; 1 >$
- To detect a CFid between a cell x (victim) and a cell y (aggressor)
  - CFid (y rise $\rightarrow$ x=0): w1x w0y w1y r1x
  - CFid (y fall $\rightarrow$ x=1): w0x w1y w0y r0x
  - CFid (y rise $\rightarrow$ x=1): w0x w0y w1y r0x
  - CFid (y fall $\rightarrow$ x=0): w1x w1y w0y r1x

CF Idempotent



Two cells with <↑;↕> CF_{in}



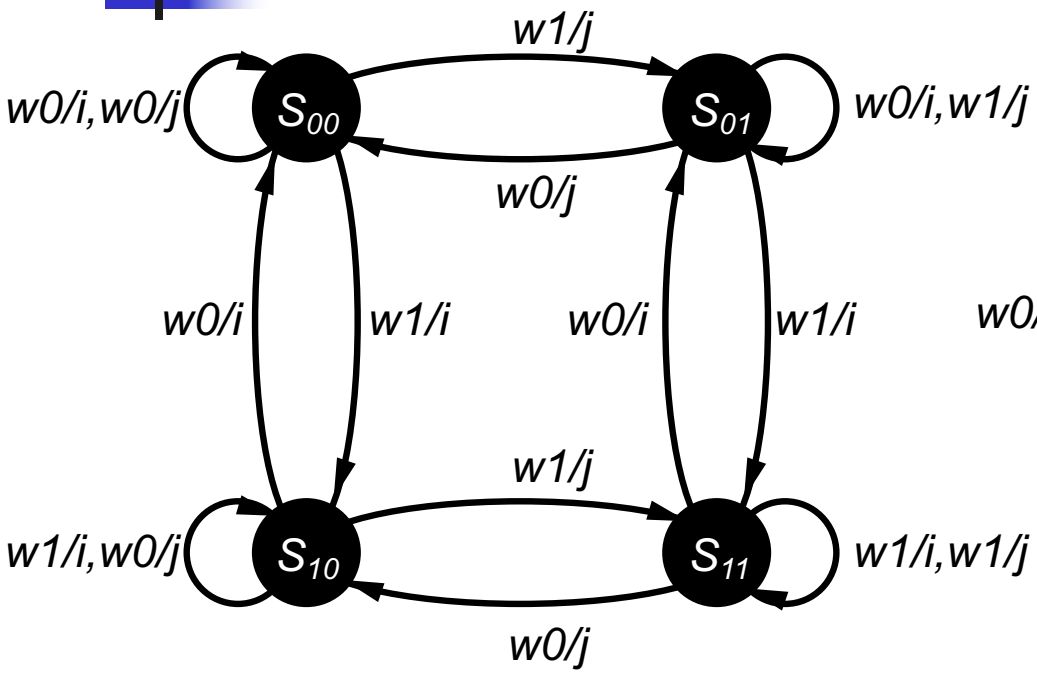
Two cells with <↑;1> CF_{id}

CF State Coupling

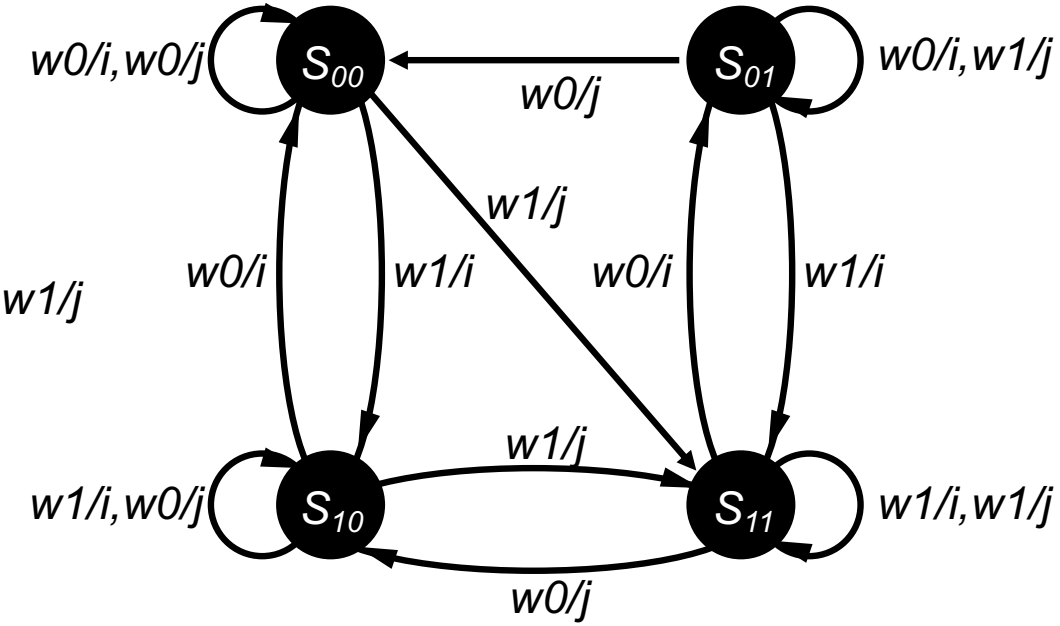
- CFst: The victim cell is forces to 0 or 1 if the aggressor cell is in a certain state
- It exists 4-types of CFst :

  $< 0 ; 0 > < 0 ; 1 > < 1 ; 0 > < 1 ; 1 >$
- To detect a CFst between a cell x (victim) and a cell y (aggressor)
  - CFst (y=0 $\rightarrow$ x=0): w1x w0y r1x
  - CFst (y=0 $\rightarrow$ x=1): w0x w0y r0x
  - CFst (y=1 $\rightarrow$ x=0): w1x w1y r1x
  - CFst (y=1 $\rightarrow$ x=1): w0x w1y r0x

CF State Coupling



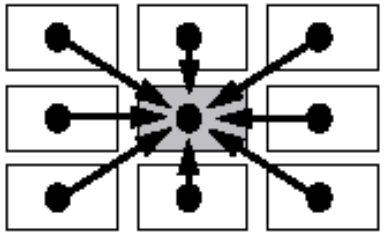
Two golden cells



Two cells with a <1;1> CFst

Pattern Sensitive (PSF)

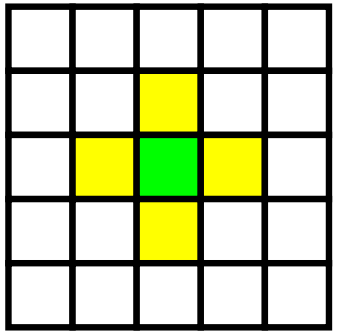
- The victim cell is forced to 0 or 1 if a certain number of neighbouring cells have a particular configuration
- PSF faults are the most general case of coupling faults
- NPSF coupling faults (Neighbourhood Pattern Sensitive Fault), reduce the neighbourhood to the cells immediately neighbouring the victim cell (called base cell)



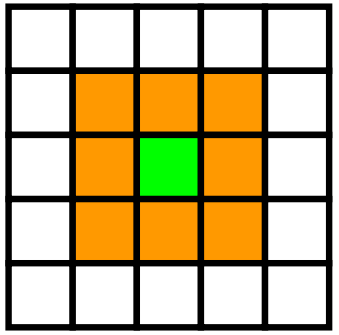
- Equivalent to a coupling fault with N aggressors (up to 8 adjacent cells
- Difficult to detect
  - For each cell, the effect of all possible combinations (28) of adjacent cells must be tested

Neighbourhood Pattern Sensitive Fault (NPSF)

- In practice two types of faults are used
  - Type-1 NPSF with 4 neighbouring cells
  - Type-2 NPSF with 8 neighbouring cells

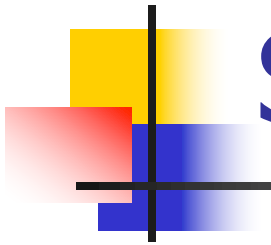


Type-1



Type-2

Neighbourhood Pattern Sensitive Fault (NPSF)



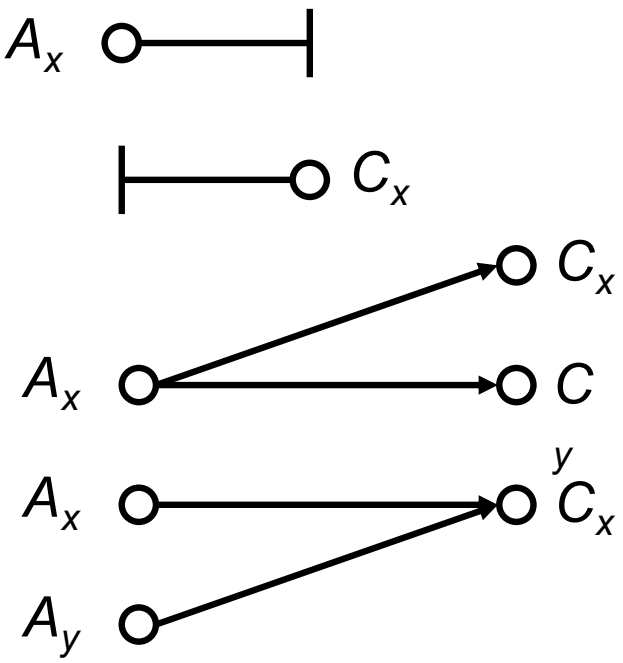
- Active NPSF (ANPSF) :
  - A change of the base-cell due to a transition in the neighbouring cells
  - Each base-cell must be read in state 0 and in state 1 for all possible changes in the configuration of neighbouring cells
- Passive NPSF (PNPSF) :
  - The change of the base-cell is impossible due to a certain configuration of the neighbouring cells
  - Each base-cell must be written and read in state 0 and in state 1 for all permutations of the configurations of neighbouring cells
- Static NPSF (SNPSF) :
  - The content of the base-cell is forced to a certain value due to a certain configuration of the neighbouring cells
  - Each base-cell must be read in state 0 and in state 1 for all permutations of the configurations of neighbouring cells

Address Decoder Faults - AF

- 4-types of address decoder faults

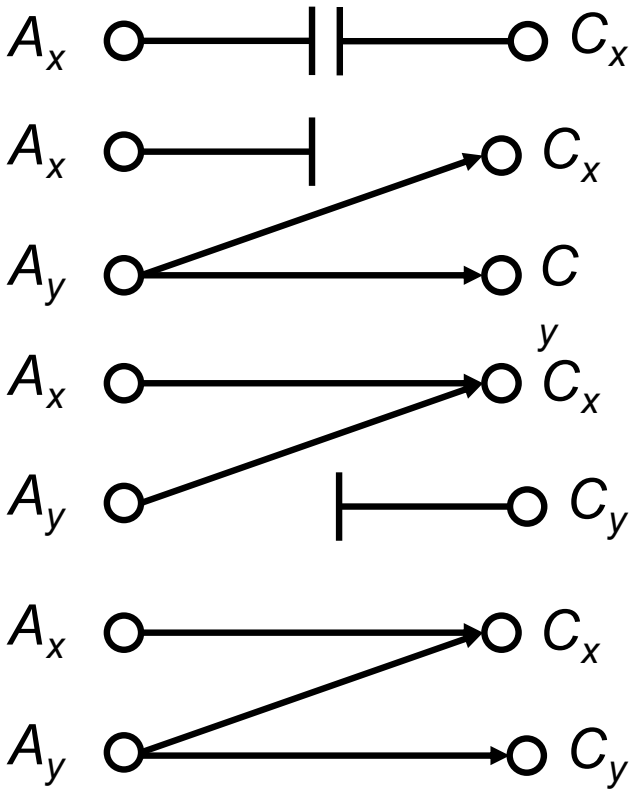
1. A certain address selects no cell
2. A certain cell is never accessed

3. A certain address selects multiple cells
4. A certain cell is accessed by multiple addresses

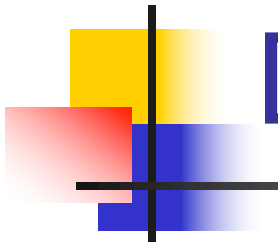


Address Decoder Faults - AF

- 4 combinations of faults for the address decoder
  - Fault A: 1+2
  - Fault B: 1+3
  - Fault C: 2+4
  - Fault D: 3+4



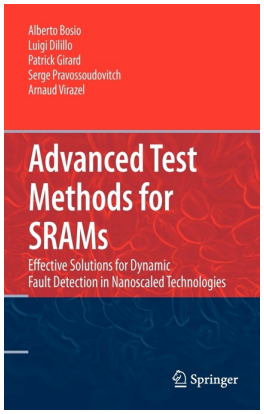
Data Retention Fault - DRF

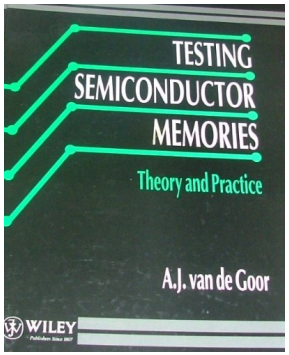


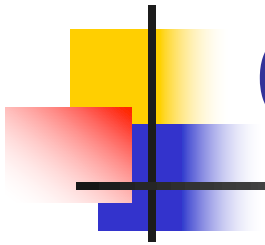
- A cell is unable to retain its value after a period of time
- Fault due for example to a faulty pull-up element in an SRAM cell
- The cell loses its value due to leakage currents
- Two different DRFs exist (loss of 1 and loss of 0) and can be simultaneously present
- To detect a DRF → One introduce a delay before reading the contents of the cell (usually ~ 10-100 ms)
- Can be very easily added to any algorithm
- The test time increases drastically!

And many others …

- Linked faults
  - Two or more faults affecting the same cell
- Dyamic Fault
  - Faults that require an ordering sequence of operations



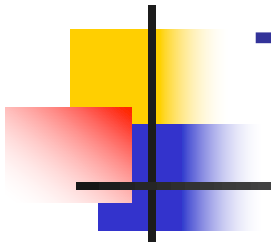




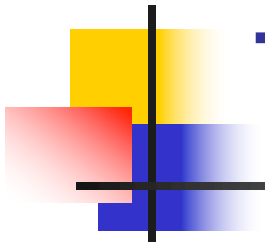
Outline

- Introduction
- Memory modelling
- Failure mechanisms and Fault Modelling
- **Test algorithms**
- Memory BIST

Test Algorithms - Notations

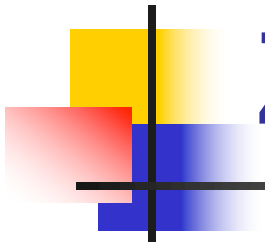


- ↑ : indicates that addresses are in an ascending order
- ↓ : indicates that addresses are in an descending order
- w0 : a write 0 at the current address
- w1 : a write 1 at the current address
- r0 : a read at the current address that must return a 0
- r1 : a read at the current address that must return a 1
- (….) : element of the algorithm
- {(…),(…),…,(…)} : full algorithm



Test Algorithms - Issues

- A full behavioural test is definitely too long
- Classical and older test methods had execution times proportional to
  - n (zero-one, checkerboard, …)
  - n^2 et $n.\log_2(n)$ (walking1/0, ping-pong, Galpat, Galcol, …)

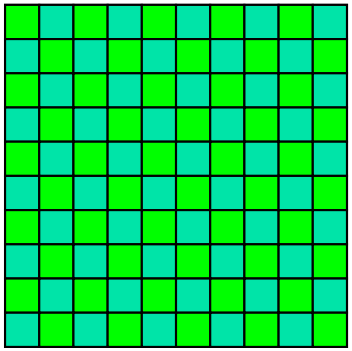


Zero-One Algorithm

- This minimal test consists of writing 0s and 1s in the memory
  - Step1: write 0 in all cells
  - Step2: read all cells (0 expected)
  - Step3: write 1 in all cells
  - Step4: read all cells (1 expected)
- O(n) test
- Coverage
  - Does not detect all Afs
  - SAFs detected if address decoder is healthy Does not detect all TFs and CFs

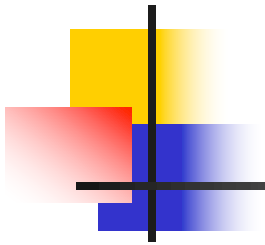
Checkerboard Algorithm

- The cells are divided into 2 groups
  - Step1: write 1 in the green cells and 0 in the blue ones
  - Step2: read (and verify) all cells
  - Step3: write 0 in the green cells and 1 in the blue ones
  - Step4: read (and verify) all cells
- O(n) test
- Coverage
  - Does not detect all AFs, TFs and CFs
  - SAFs detected if address decoder is healthy
  - This test is able to detect short-circuit faults



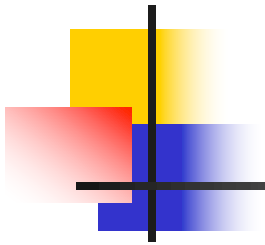
Test Application Time

	Number of Operations		
Size	n	$n.\log_2 n$	n^2
1Mb	0.063	1.26	18.33
16Mb	1.01	24.16	4691.3
256Mb	16.11	451	1200959.9
2Gb	128.9	3994.4	76861433.7



March Test

- The test walks (« March ») through the memory
- The test is composed of March elements which are represented between ()
- March tests are the simplest tests to detect SAFs, TFs and CFs



Example - MATS++

{↑(w0); ↑(r0,w1); ↓(r1,w0,r0)}

- 6n operations

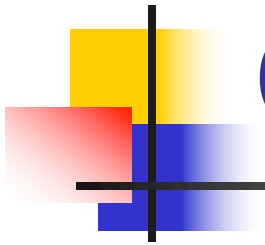
- Detect al SAFs
- Detect all Afs
- Detect all TFs

Some March Tests

Name	Ref.	algorithm
MATS	[NAI79]	{ |(w0);| (r0,w1);|(r1)}
MATS+	[ABA83]	{|(w0);⇑(r0,w1); ⇓(r1,w0)}
MATS++	[VAN91]	{|(w0);⇑(r0,w1);⇓(r1,w0,r0)}
March X	[VAN91]	{|(w0);⇑(r0,w1);⇓(r1,w0);|(r0)}
March C-	[MAR82]	{|(w0);⇑(r0,w1);⇑(r1,w0);⇓(r0,w1);⇓(r1,w0);|(r0)}
March A	[SUK81]	{|(w0);⇑(r0,w1,w0,w1);⇑(r1,w0,w1);⇓(r1,w0,w1,w0);⇓(r0,w1,w0)}
March Y	[VAN91]	{|(w0);⇑(r0,w1,r1);⇓(r1,w0,r0);|(r0)}
March B	[SUK81]	{|(w0);⇑(r0,w1,r1,w0,r0,w1);⇑(r1,w0,w1);⇓(r1,w0,w1,w0); ⇓(r0,w1,w0)}
March GS	[VAN93]	{|(w0);⇑(r0,w1,r1,w0,w1);⇑(r1,w0,r0,w1);⇓(r1,w0,w1,w0); ⇓(r0,w1,r1,w0);Del;|(r0,w1,r1);Del;|(r1,w0,r0)}
March M	[MIK96]	{|(w0);⇑(r0,w1,r1,w0);|(r0);⇑(r0,w1);|(r1);⇑(r1,w0,r0,w1,);|(r1);⇓(r1,w0)}
March LR	[VAN96]	{|(w0);⇓(r0,w1);⇑(r1,w0,r0,w1);⇑(r1,w0);⇑(r0,w1,r1,w0,);⇑(r0)}
March U	[VAN97]	{|(w0);⇑(r0,w1,w0,w1,r1);⇑(r1,w0,w1,w0,r0);⇓(r0,w1,w0,w1,r1);⇓ (r1,w0,w1,w0,r0; ⇓ (r0)}
March LA	[VAN99]	{|(w0);⇑(r0,w1,r1,w0);⇑(r0,w1); ⇓ (r1,w0,r0,w1); ⇓ (r1,w0)}
March SR	[VAN00]	{⇓ (w0);⇑(r0,w1,r1,w0);⇑(r0,r0););⇑(w1),⇓ (r1,w0,r0,w1); ⇓ (r1,r1)}
March SS	[HAM02]	{|(w0);⇑(r0,r0,w0,r0,w1);⇑(r1,r1,w1,r1,w0);⇓(r0,r0,w0,r0,w1);⇓ (r1,r1,w1,r1,w0; |(r0)}

Fault coverage

March	Fault coverage								#Op.
	SAF	AF	TF	CFin	CFid	CFdyn	SCF	Linked Faults	
MATS	All	Some							4.n
MATS+	All	All							5.n
MATS++	All	All	All						6.n
March X	All	All	All	All					6.n
March C-	All	All	All	All	All	All	All		10.n
March A	All	All	All	All				All linked CFids, some CFins linked with CFids	15.n
March Y	All	All	All	All				All TFs linked with CFins	8.n
March B	All	All	All	All				All linked CFids, all TFs linked with CFids or CFins, some CFins linked with CFids	17.n

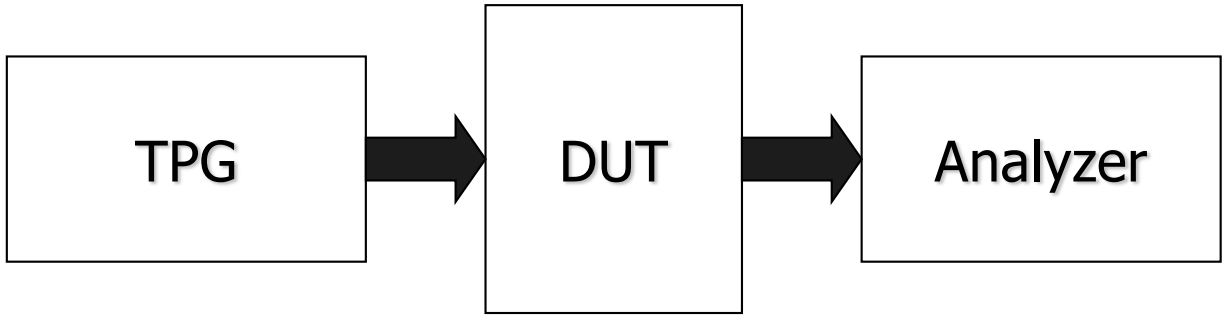


Outline

- Introduction
- Memory modelling
- Failure mechanisms and Fault Modelling
- Test algorithms
- **Memory BIST**

BIST Principle

- In BIST (Built-In Self-Test) techniques, test vectors are produced and the results analysed on the circuit



- The objective is to solve all or part of the following problems
  - poor controllability and poor observability
  - test at nominal operating speed
  - cost of the ATE
  - reduction of test time
  - "transparent" test for designers, etc.

MBIST Principle