EEDG/CE 6303: Testing and Testable Design

Mehrdad Nourani

Dept. of ECE
Univ. of Texas at Dallas

Session 08

Memory Testing

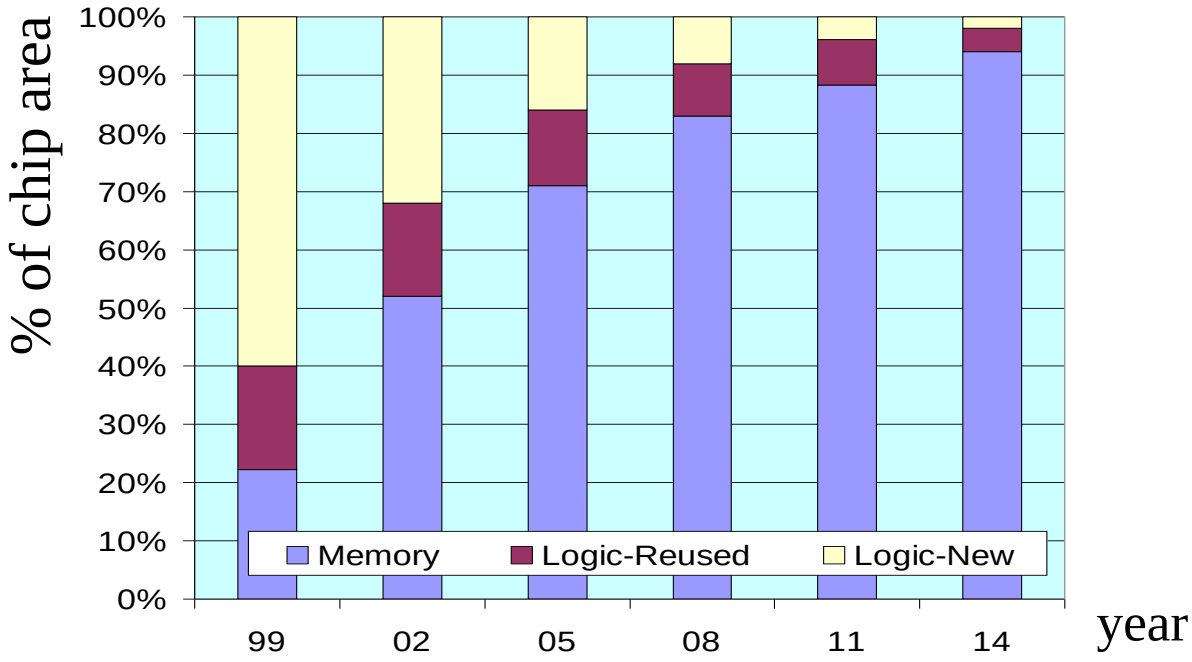
Key Issues

- Motivation for testing memories
- Modeling memory chips
- Reduced functional fault models
- Traditional tests
- March tests
- Pseudorandom memory tests

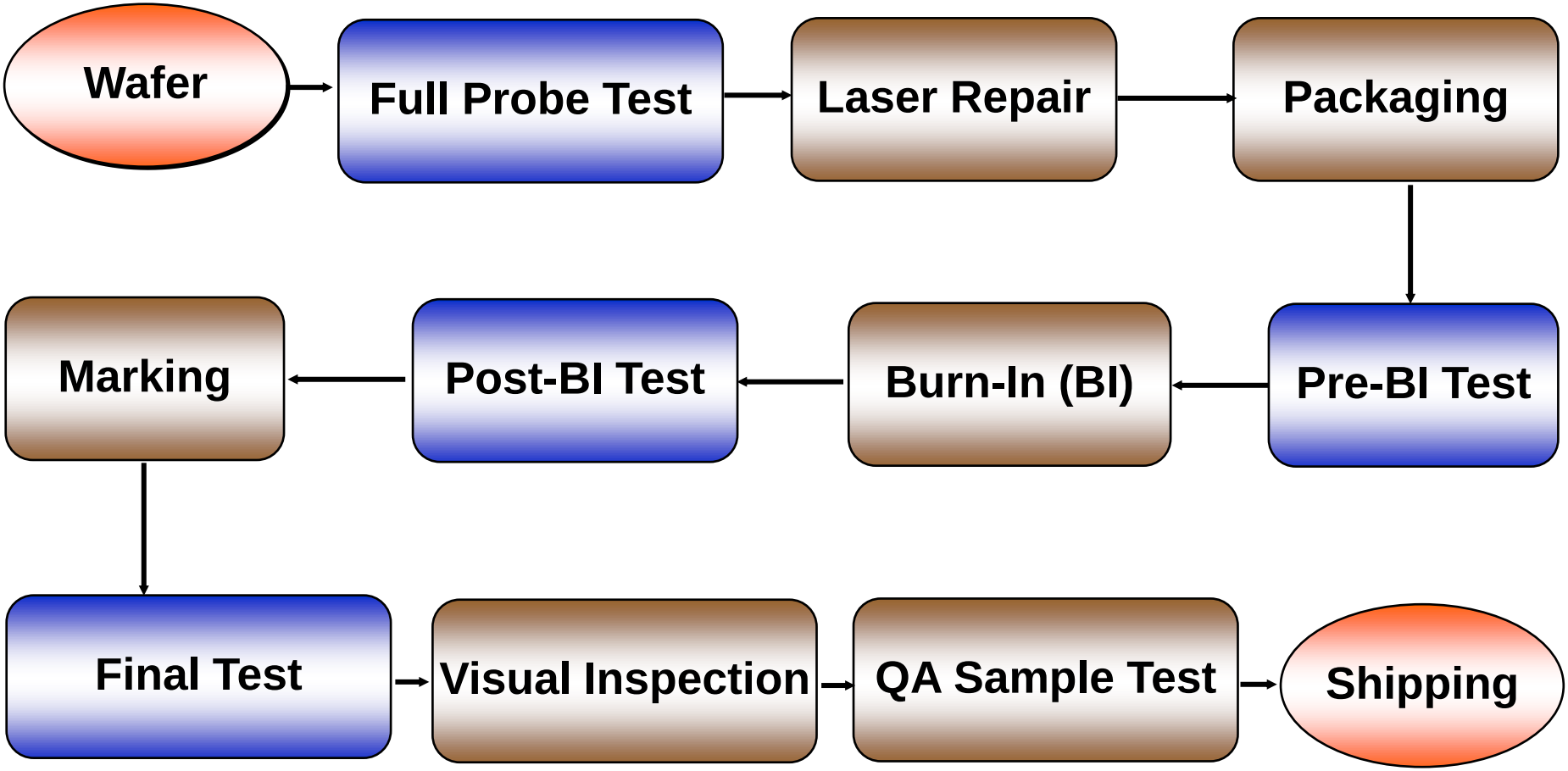
Motivation for Memory Testing

Importance of Memories

- Memories dominate chip area (avg. 94% of chip area in 2014)
- Memories are most defect sensitive parts
  - Because they are fabricated with minimal feature widths
- Memories have a large impact on total chip DPM level
  - Therefore high quality tests required
- (Self) Repair becoming standard for larger memories (> 1 Mbit)

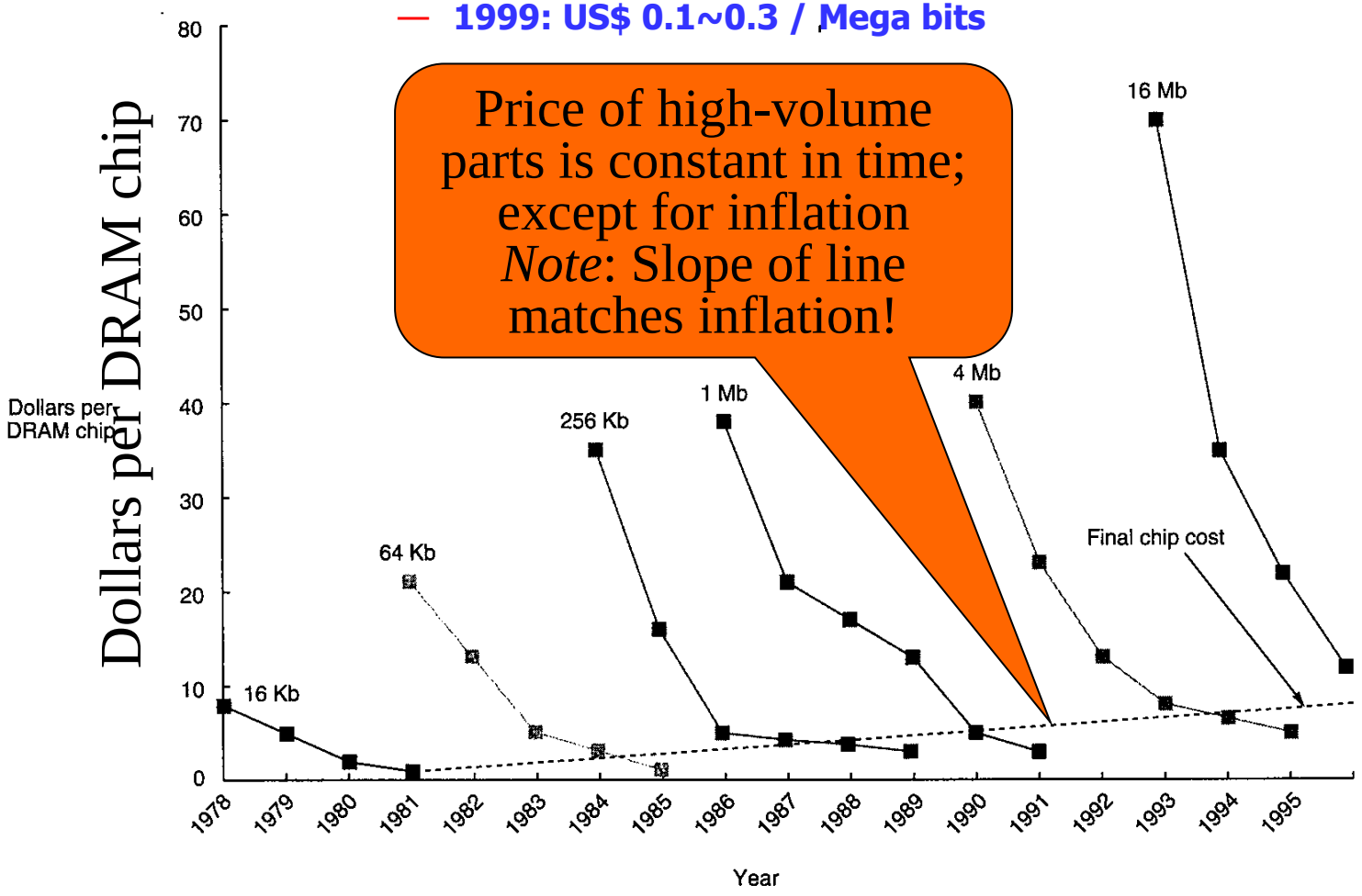


Typical RAM Production Flow



Memory Chip Cost Over Time

- **DRAM Price per Bit**
  - **1991: US$ 40 / Mega bits**
  - **1995: US$ 3.75 / Mega bits**
  - **1999: US$ 0.1~0.3 / Mega bits**



Complexity of Memory Test

n Number of bits	Algorithm complexity (Cycle time = 150 ns)			
	$O(n)$	$O(n*\log_2 n)$	$O(n^{3/2})$	$O(n^2)$
1 K	0.0001 s	0.001 s	0.003 s	0.1 s
1 M	0.1 s	2.1 s	110 s	30.6 h
4 M	0.42 s	9.2 s	860 s	20.3 d
16 M	1.68 s	40.3 s	1.91 h	325 d
64 M	6.71 s	174 s	15.3 h	14.2 y
256 M	26.8 s	2.09 h	122 h	228 y
1 G	107 s	8.94 h	977 h	3655 y

- A memory is tested with *several algorithms*, which together may go through the total address space over 100 times
- Effective test time for 16 Mb DRAM, using $O(n)$ tests, is about 1.68s
  - Test time reduced using on-chip parallelism (DFT and BIST)

Fault Types (Review)

- 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

Fault Types (cont.)

- **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)
- **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)
- **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)

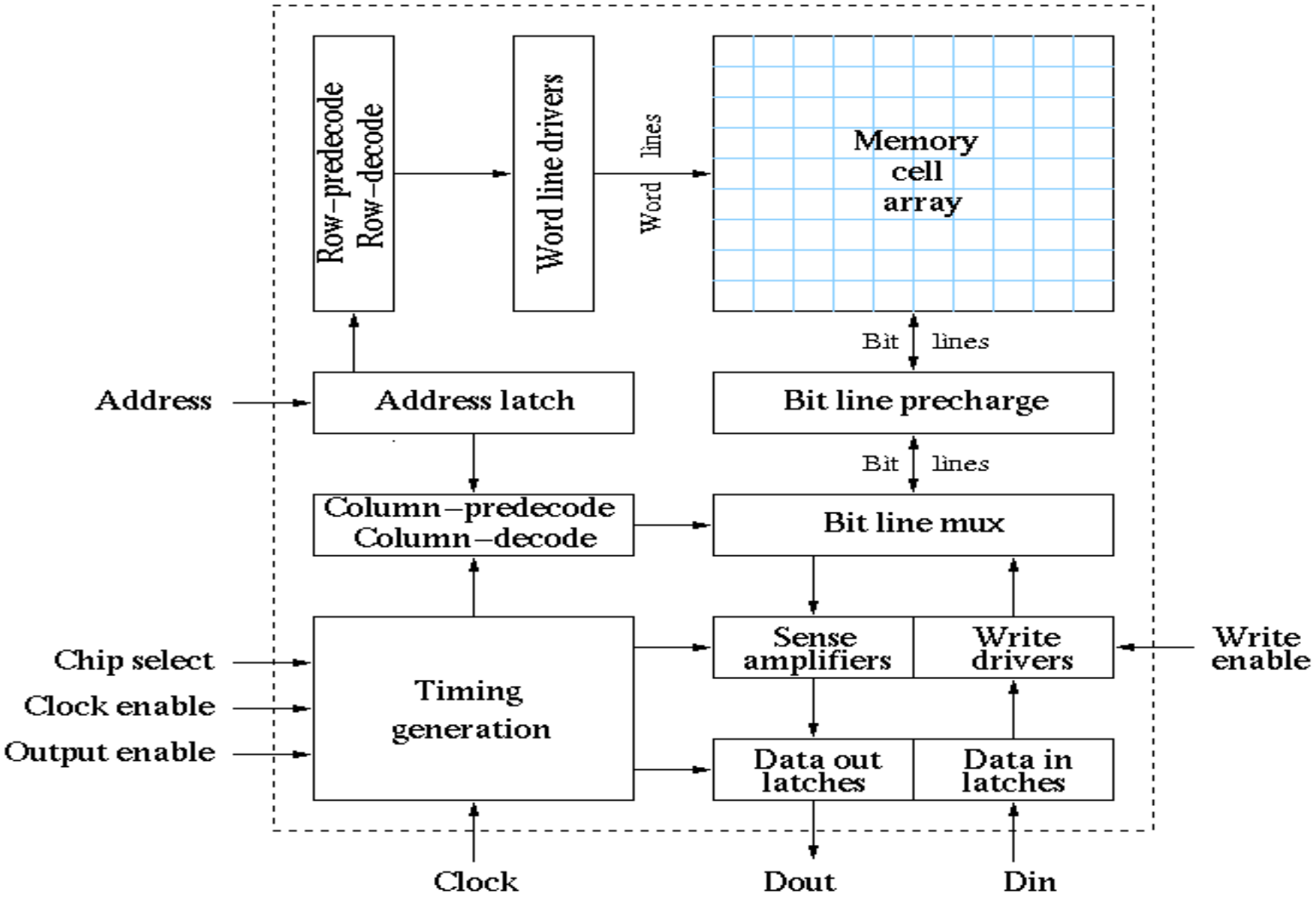
Challenges in Testing Memories

- Number of bits/chip increases exponentially (4x in <3 years)
- Price/bit decreases exponentially
- Test cost has to decrease exponentially as well
  - Older traditional tests were of O(n^2)
  - Current tests have to be of O(n) or less
    - DFT and BIST used to reduce test time to about O(n)
- Area/cell decreases exponentially
  - Line widths decrease (more: opens, resistance)
  - Line distances decrease (more: shorts, couplings)
  - Cell leakage increases with reduced threshold voltages
- *Consequence*
  - More complex fault behavior
  - More global fault behavior (ground bounce, coupling effects)

- **Result:** *Number of bits to be tested increases exponentially, fault behavior becomes more complex, while test cost has to be same*

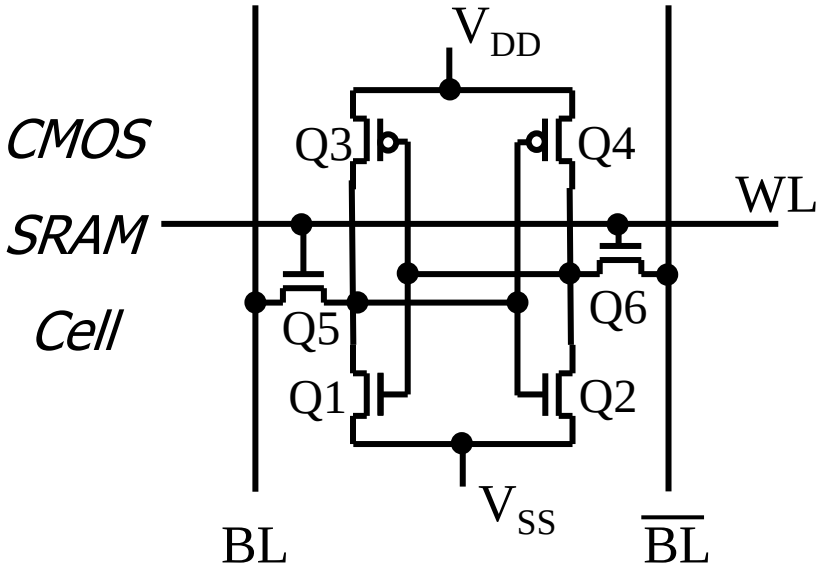
Modeling Memory Chips

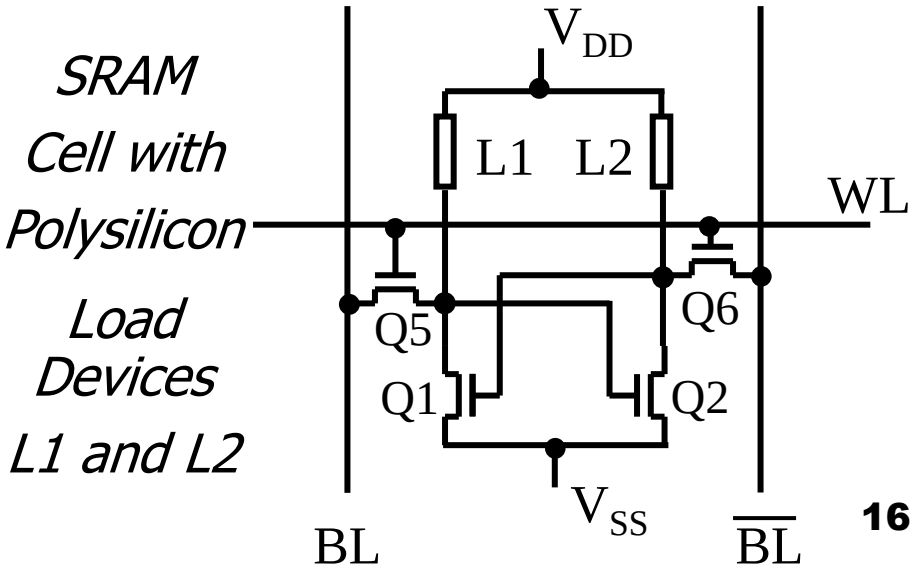
Functional SRAM Memory Model



Electrical SRAM Cell Model

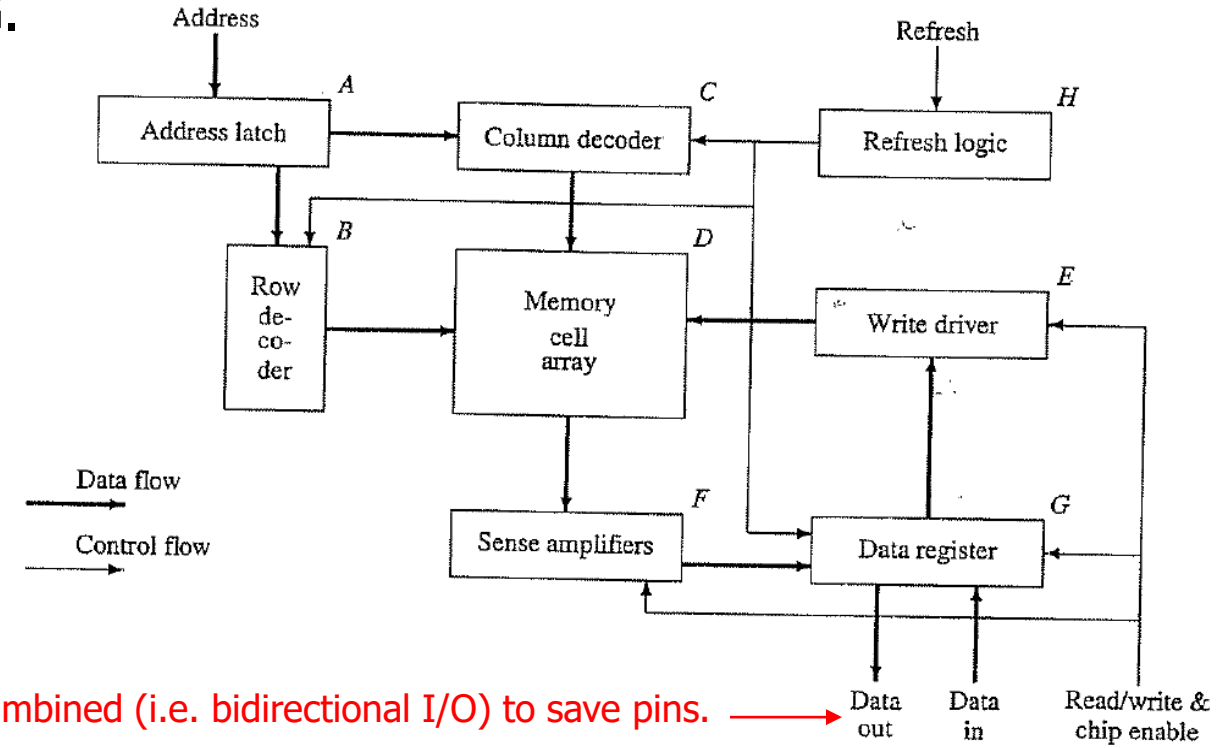
- Structure of a CMOS SRAM cell (left cell below)
  - Q1--Q3 and Q2--Q4 form inverters (Q3 and Q4 are the load devices)
  - Inverters are cross coupled to form a latch
  - Q5 and Q6 are pass transistors
- Addressing of a cell uses
  - a row address (using the *word line* WL)
  - a column address (using the *bit lines* BL and BL')
- Operations
  - *Write*: precharge BLs; drive BL, BL' and WL
  - *Read*: precharge BLs; drive WL; feed BLs to sense amplifiers





Functional DRAM Memory Model

- **Read**: contents of the selected cells are amplified by the sense amplifiers F, loaded into the data register G and presented on the data-out lines.
- **Write**: the data on the data-in lines are loaded into the data register and written into the memory cells through the driver E.
- **Refresh**: the column decoder selects all the columns, the row decoder selects a row by the row decoder, all bits in the selected row are read and written (refreshed) simultaneously. During refresh, block H disables the data register G.



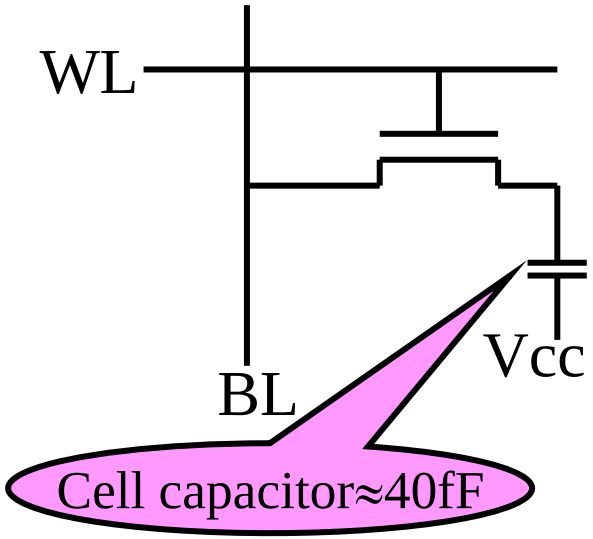
Often combined (i.e. bidirectional I/O) to save pins.

Electrical DRAM Cell Model

DRAM cell stores information as a *charge* in a capacitor

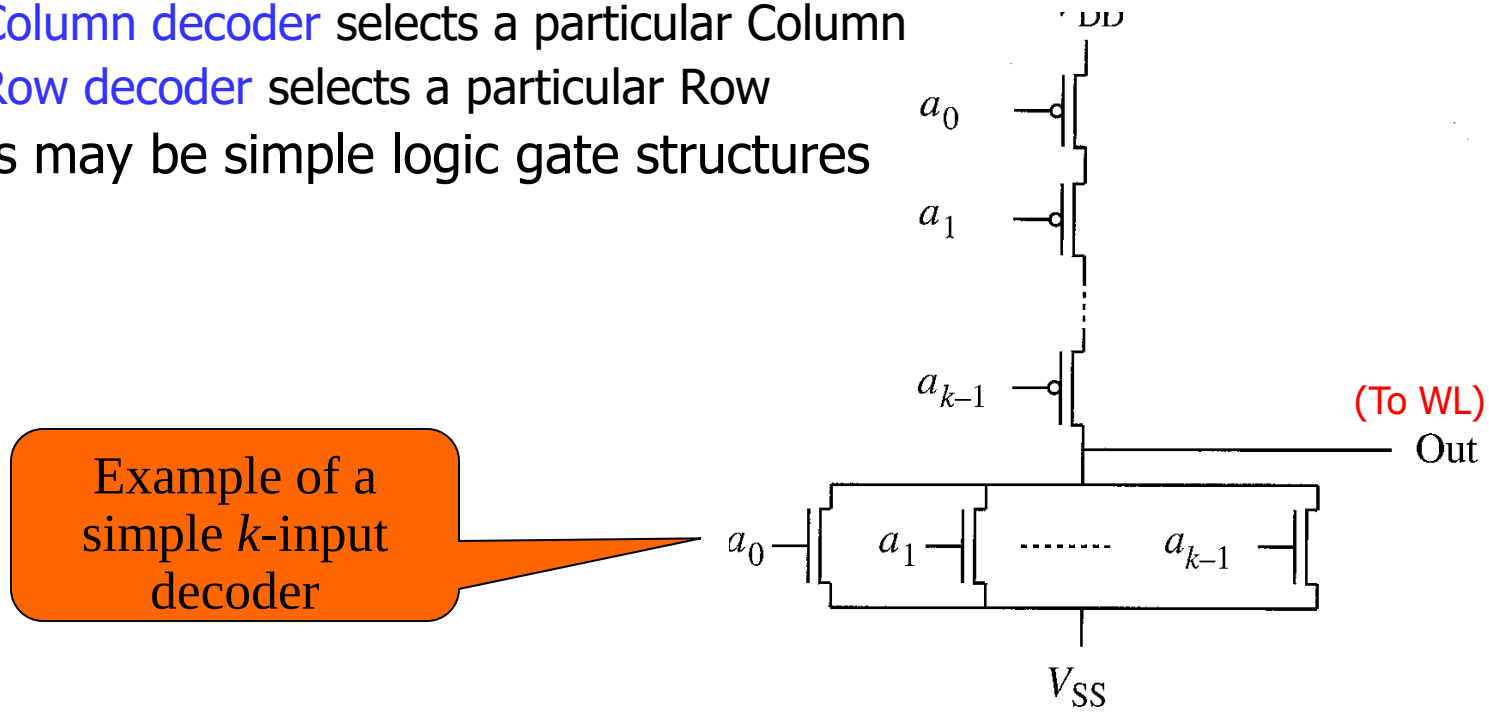
- Cell capacitance ≈ 40 fF = 40* 10^{-15} F; BL capacitance ≈ 300 fF!
  (SRAM cell stores information in terms of the *state of a latch*)
- This charge leaks away over time

- DRAMs require refresh circuitry (Refresh rate: ≈ 64 ms)
  - DRAM cells require 1/4th the area of SRAM cells
  - DRAM cells dissipate less power (thicker gate oxide)
  - DRAMs are less sensitive to Soft Errors
  - DRAMs are slower
- Operations
  - *Read*: precharge BL; drive WL;
    feed BL to sense amplifier
  - *Write*: drive BL; drive WL



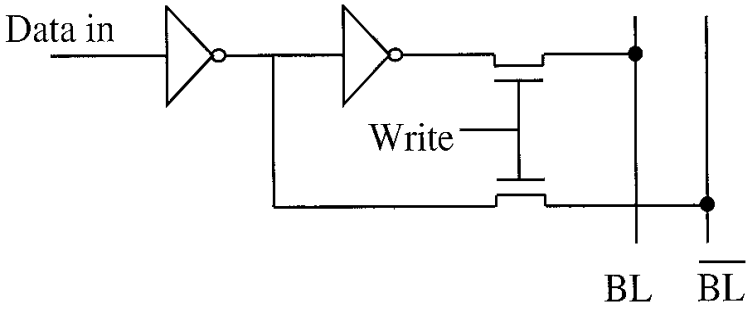
Decoders

- Decoders address a specific cell in the Memory Cell Array 'MCA'
- If the MCA would be a *vector*, then 1 Mbit MCA requires 1M WLs
- By arranging the cells in the MCA in a two-dimensional structure, a 1 Mbit MCA requires 1K WLs and 1 K BLs
  - This is a significant reduction in the decoder area!
- Hence, the MCA is two dimensional and consists of rows and columns
  - The Column decoder selects a particular Column
  - The Row decoder selects a particular Row
- Decoders may be simple logic gate structures



Read-Write Drivers

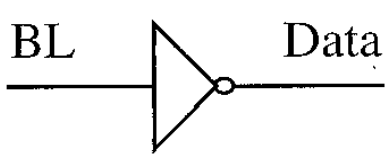
- The write driver can be rather simple
  - The data-to-be-written is presented on BL
  - The inverse data-to-be-written is presented on BL'



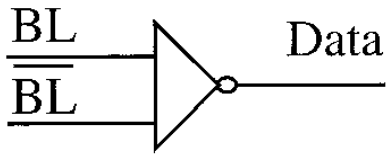
- The sense amplifier can be

  a. a simple inverter; e.g. for small arrays, which have strong signals

  b. a differential amplifier; for larger arrays, which have weak signals



(a) Inverter



(b) Differential amplifier

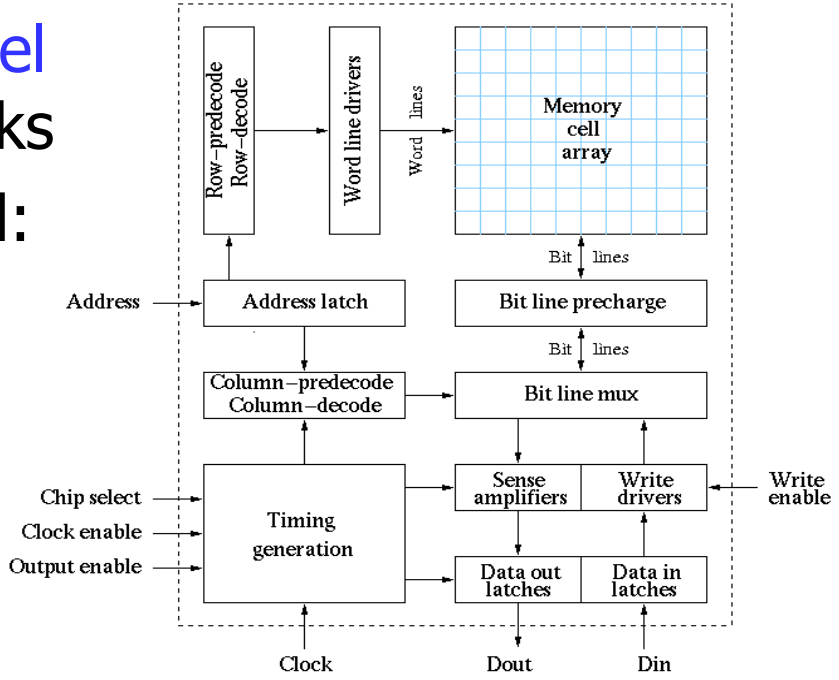
A Subset of Functional Faults

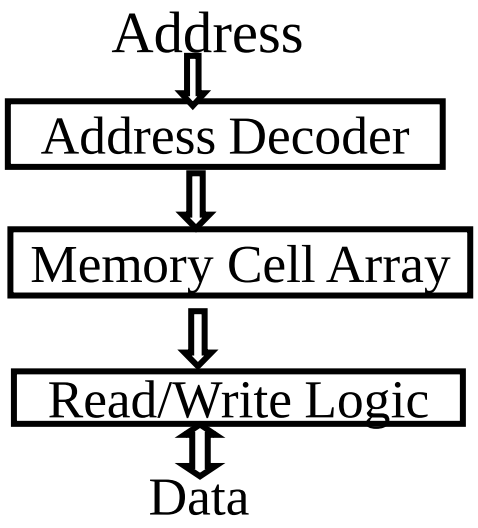
- a. Cell stuck
- b. Driver stuck
- c. Read/write line stuck
- d. Chip-select line stuck
- e. Data line stuck
- f. Open circuit in data line
- g. Short circuit between data lines
- h. Crosstalk between data lines
- i. Address line stuck
- j. Open circuit in address line
- k. Shorts between address lines
- 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 Fault Models

The Reduced Functional Model

- For test purposes Functional model reduced (simplified) to *three* blocks
- The functional faults also reduced:
  - —single-cell faults (SAFs, TFs, etc.)
  - —two-cell faults (Coupling faults)
  - —*k*-cell faults (NPSFs)
- The functional fault models have a *hierarchy*:
  1. Stuck-at fault (SAF)
  2. Transition fault (TF)
  3a. Coupling fault (CF)
  3b. Neighborhood pattern sensitive fault (NPSF)





Notations for Describing Faults

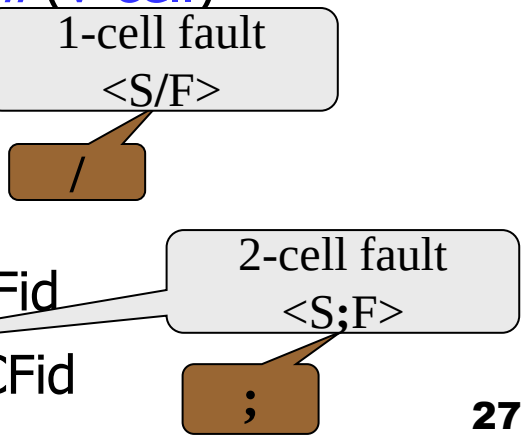
- r -- Read a memory location
- w -- Write a memory location
- r0 -- Read a 0 from a memory location
- r1 -- Read a 1 from a memory location
- w0 -- Write a 0 to a memory location
- w1 -- Write a 1 to a memory location
- ↑ -- Write a 1 to a cell containing 0
- ↓ -- Write a 0 to a cell containing 1
-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

Notations for Describing Faults (cont.)

- <…> describes a fault
- <*S*/*F*> describes a *single-cell fault*
  - *S* describes the state/operation *sensitizing* the fault
    - A fault is sensitized when the fault effect is made present
  - *F* describes the *fault effect* in the *cell*
- <*S*;*F*> describes a *two-cell fault* (a Coupling Fault)
  - *S* describes the state/operation of the *aggressor cell* (a-cell) sensitizing the fault
  - *F* describes the *fault effect* in the *victim cell* (v-cell)

Examples:

- <∀/0>: a SA0 fault
- <∀/1>: a SA1 fault
- <↑/0>: a ↑ TF
- <↓/1>: a ↓ TF
- <↑;0>=<↑;↓>: a CFid
- <↑;1>=<↑;↑>: a CFid
- <↓;0>=<↓;↓>: a CFid
- <↓;1>=<↓;↑>: a CFid

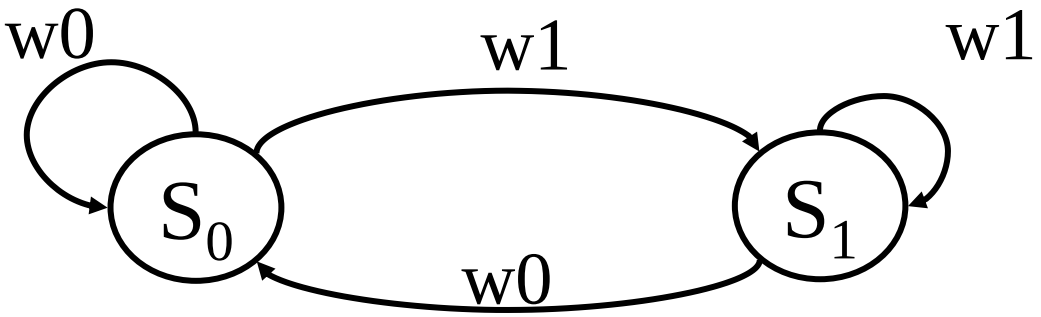


Stuck-at-Fault (SAF)

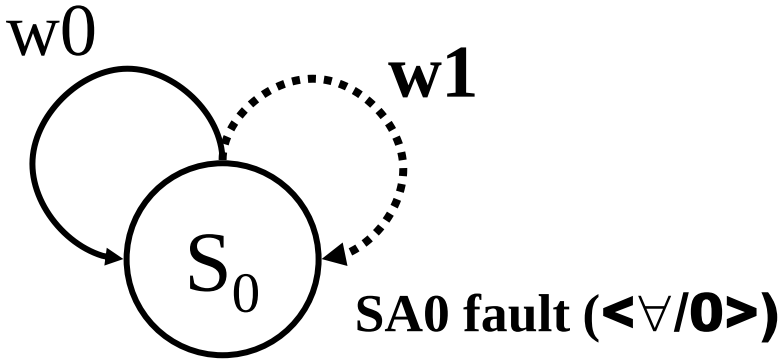
- The logic value of a stuck-at cell or line is always:
  - 0: a SA0 fault
  - 1: a SA1 fault

  } SAF *fault type* has two *subtypes*

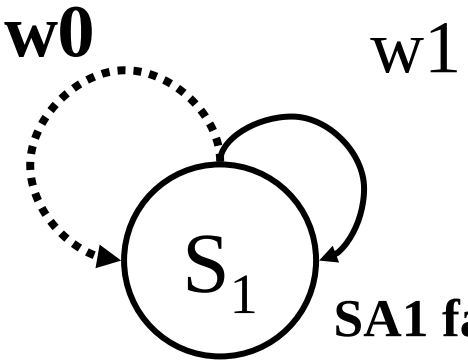
- Only *one* fault (a SA0 or a SA1) can be present at a time



Good cell



SA0 fault (<∀/0>)



SA1 fault (<∀/1>)

Stuck-Open Fault (SOpF)

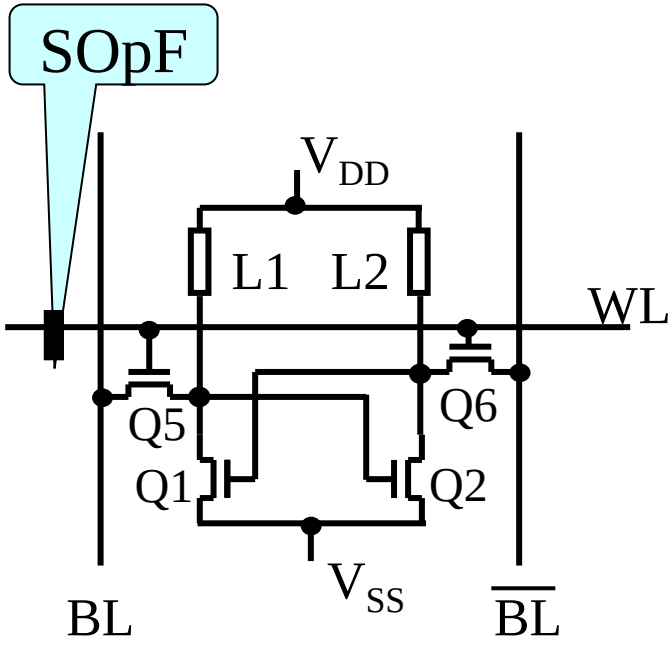
- SOpF: A cell cannot be accessed; e.g., due to an open in its WL
- When a read operation is applied to a cell, the differential sense amplifier has to sense a voltage difference between BL and BL'
- In case of an SOpF BL and BL' have the same (high) level

Output Sense Amplifier will be one of these:

1. **fixed value** (SOpF behaves as a SAF)
2. **previous value of sense amplifier**

   Detected when in a march element a '0' and '1' is read. Required form of the march element: (…,r*x*,…,r*x**)
3. **random** (fault *detected probabilistically*)

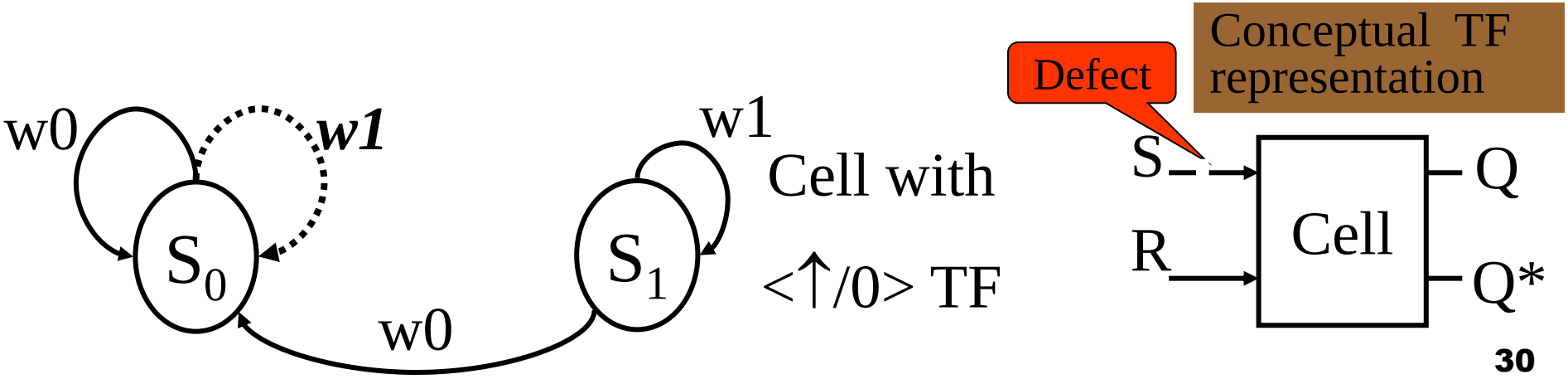
Every read operation detects the SOpF with a probability of 50%



Transition Fault

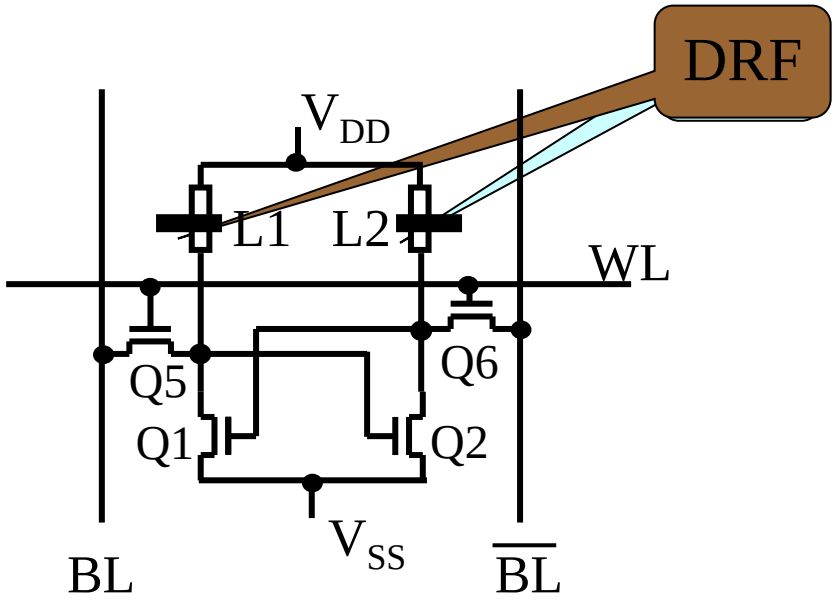
- A cell fails to undergo the following transition(s):
  - *Fault type* has two *subtypes*: the <↑/0> and the <↓/1> TF
  - Fails up-transition: a <↑/0> TF
  - Fails down-transition: a <↓/1> TF
- A single cell may contain *both* faults
  - The *Set* input may not work (e.g., SA0)
  - The *Reset* input may not work

Test: Every cell should make ↑ and ↓ transition and be read



Data Retention Fault (DRF)

- DRF: A cell cannot retain its logic value
- Typically caused by a broken pull-up device which causes the leakage current of the node with a logic 1 not to be replenished
- After a time delay '*Del*' (Typically: 100 ms ≤ *Del* ≤ 500 ms) the cell will flip
- The DRF fault type has two subtypes, which may be present simultaneously in the same cell: $<1_T/0>$ and $<0_T/1>$
  - $<1_T/0>$ means that a logic 1 will become a logic 0 after a delay time *T*

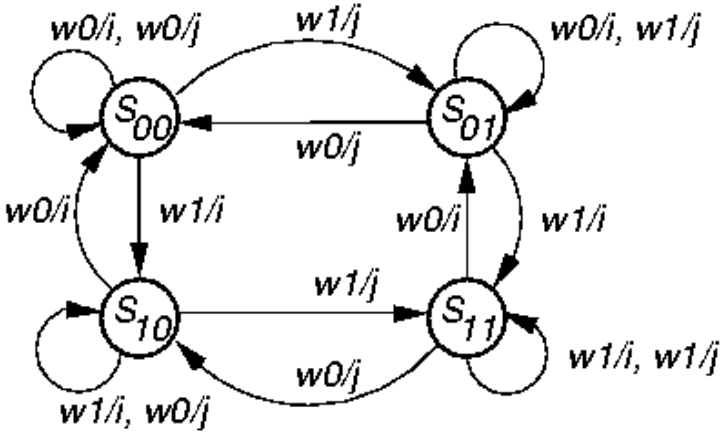


Faults Involving 2 Cells: Coupling Faults (CF)

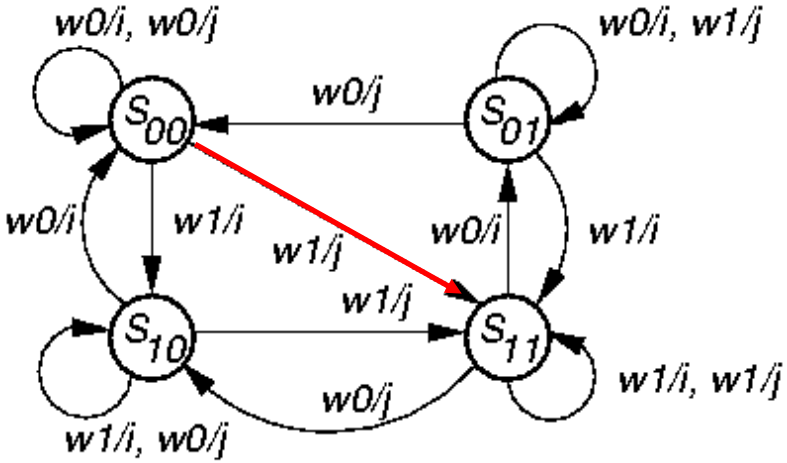
- CF: The state of, or an operation applied to, the *a-cell* (*aggressor* cell) forces or changes the state of the *v-cell* (*victim* cell)
- One example of coupling fault is *Idempotent CF* : CFid
  - A *transition write operation* ($\uparrow$=0 $\rightarrow$1 & $\downarrow$=1$\rightarrow$0 change) applied to the a-cell forces the contents of the v-cell
  - CFid has *4 subtypes*: <$\uparrow$;0>, <$\uparrow$;1>, <$\downarrow$;0> and <$\downarrow$;1> (also shown as <$\uparrow$;$\downarrow$>, <$\uparrow$;$\uparrow$>, <$\downarrow$;$\downarrow$>, <$\downarrow$;$\uparrow$>, respectively)
    - The $\uparrow$or $\downarrow$ write operations to a-cell sensitize the fault
    - The '0' or '1' are the fault effect in the v-cell
  - In addition, each subtype has *two positions*:
    A. address of a-cell < address of v-cell
    B. address of a-cell > address of v-cell

Coupling Faults (CFid)

- Idempotent CF (CFid): The ↑or ↓ transitions on coupling (aggressor) cell j sets the coupled (victim) cell i to 0 or 1.
- *Four* fault subtypes : <↑;0>=<↑;↓>, <↑;1>=<↑;↑>, <↓;0>=<↓;↓>, <↓;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



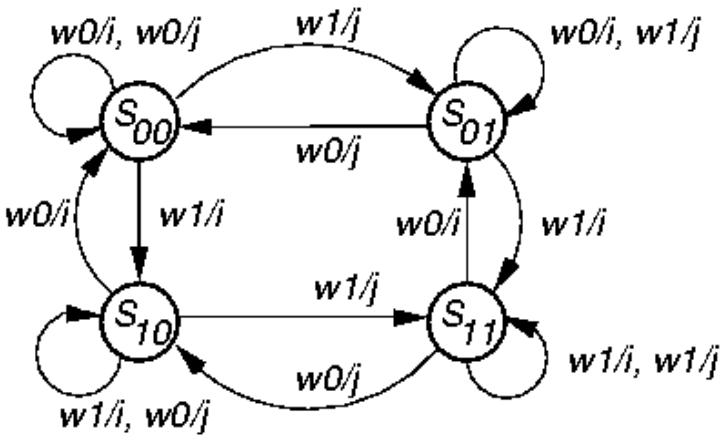
State transitions of two good (fault-free) memory cells i & j



(c) State diagram of an <↑;1 > CFid.
(aggressor: cell j ; victim: cell i)

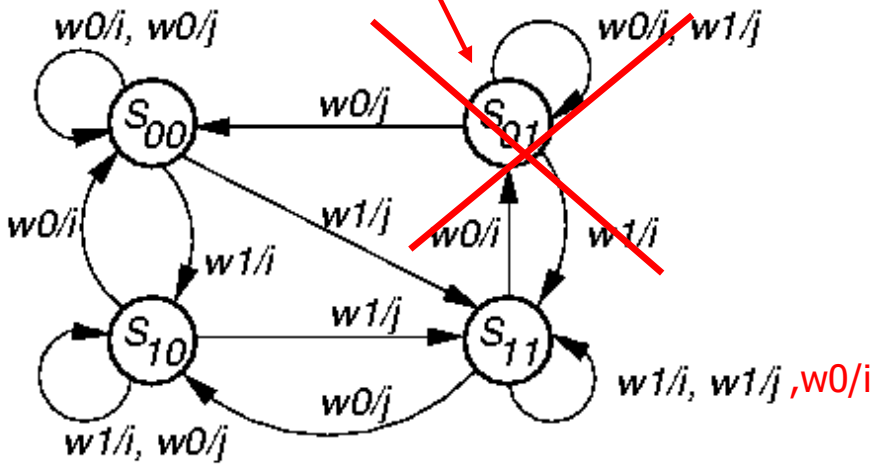
Coupling Faults (CFst)

- State CF (CFst): A CF whereby the *state of the a-cell* forces a fixed value in the v-cell
  - *Four* fault subtypes : <1;0>, <1;1>, <0;0> and <0;1>
  - Each subtype has *2* positions:
    - addr. a-cell < addr. v-cell
    - addr. a-cell > addr. v-cell



State transitions of two good (fault-free) memory cells i & j

Note: If cell i (victim) remains in "1" as long as cell j (aggressor) is in state "1", then we won't have S01 at all.

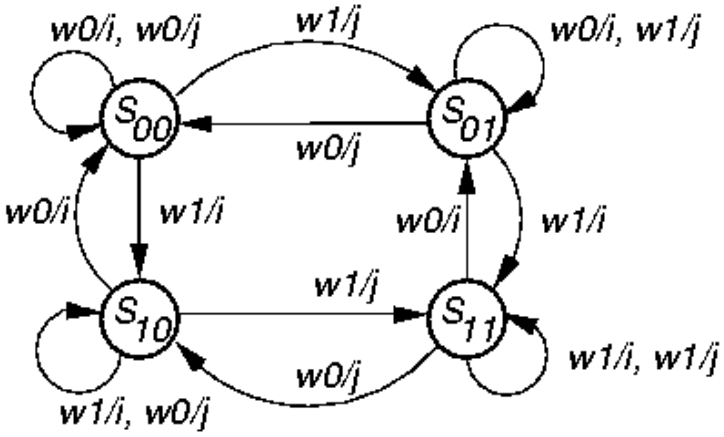


(b) Diagram of a state coupling fault (SCF) <1; 1>.
(aggressor: cell j ; victim: cell i)

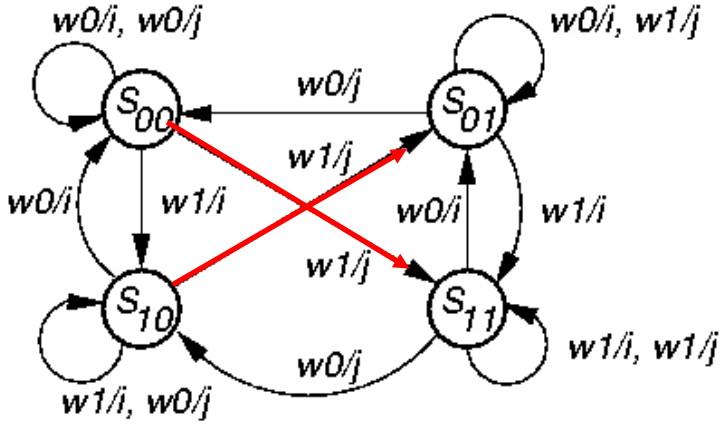
Coupling Faults (CFin)

- Inversion CF (CFin): A *transition write operation* to the a-cell *toggles* the contents of the v-cell (*Note*: ↕ denotes toggling)
  - *Two* fault subtypes exist: <↑;↕> and <↓;↕>
  - Each subtype has *2* positions:
    - addr. a-cell > addr. v-cell
    - addr. a-cell < addr. v-cell

Note: CFin is not a *realistic* fault
An SRAM cell behaves like a *latch*, rather then a *flip-flop*
⇒ *Cannot toggle!*



State transitions of two good (fault-free) memory cells i & j

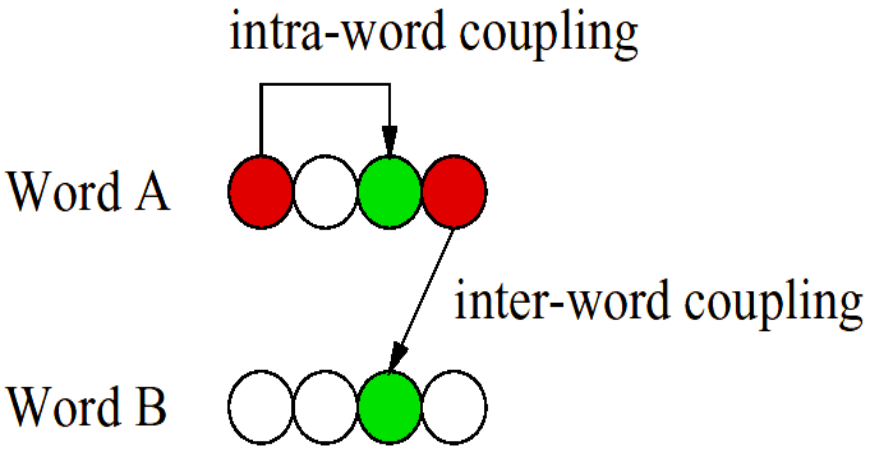


(b) State diagram of an <↑;↕> CFin.

(aggressor: cell j ; victim: cell i)

Dynamic Coupling Faults

- 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
- Intra- vs. Inter-word coupling:



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>

Pattern Sensitive Fault (*k*-CF, PSF)

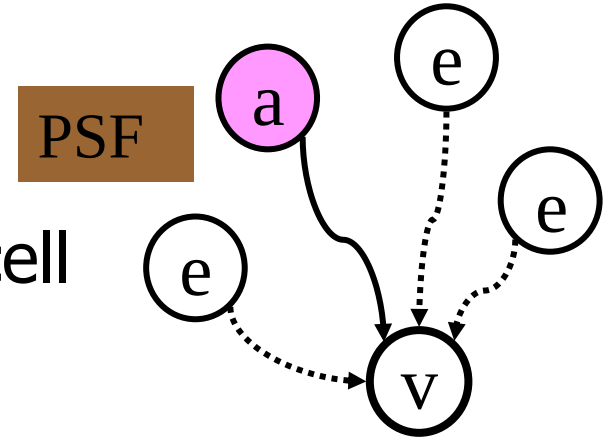
- A *k-Coupling Fault* (*k-CF*), also called a *Pattern Sensitive Fault* (*PSF*), involves *k* cells which form a *neighbourhood*
  - The v-cell is also called the *base cell*
  - The *k*-1 non-v-cells are the *deleted neighbourhood cells*
  - Any combination of aggressors/enabling cells allowed
  - The *k* cells can be *anywhere* in memory

Pattern Sensitive Fault (*cont.*)

- Example: Active PSF

  A CFid with *k-2* enabling values

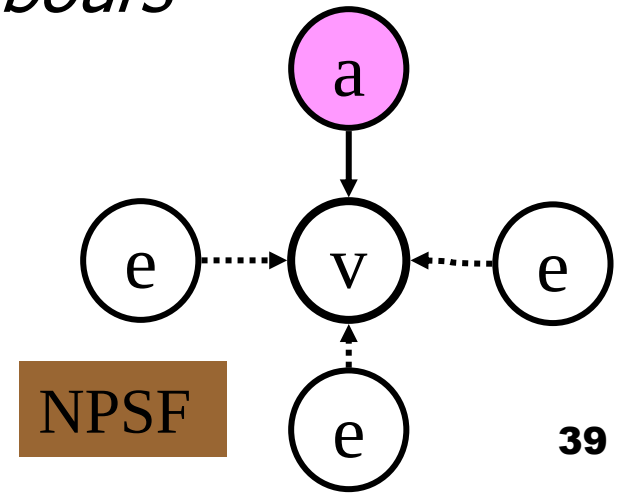
  a = a-cell, v = v-cell, e = *enabling* cell



- *Neighbourhood* PSFs (*NPSF*s) more realistic
  - —The cells have to be *physical neighbours*
  - —Example: Active NPSF (ANPSF)

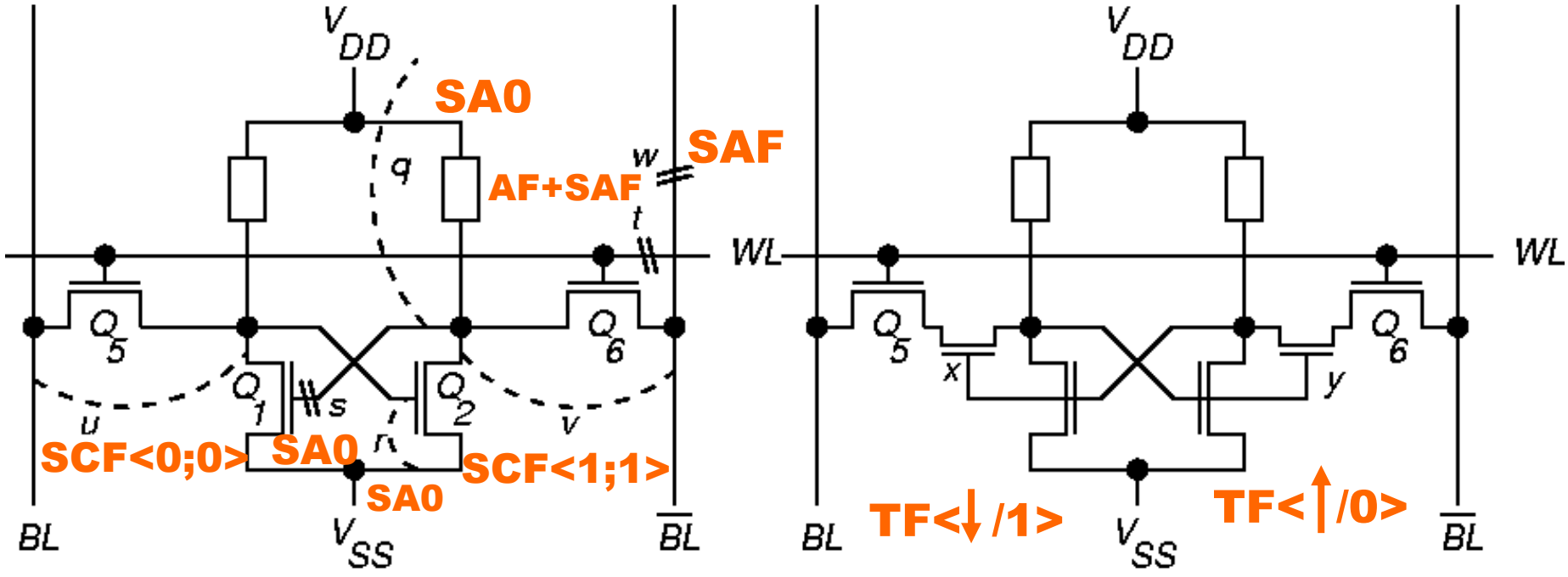
    a-cell sensitises fault in v-cell

    iff e-cells have enabling state

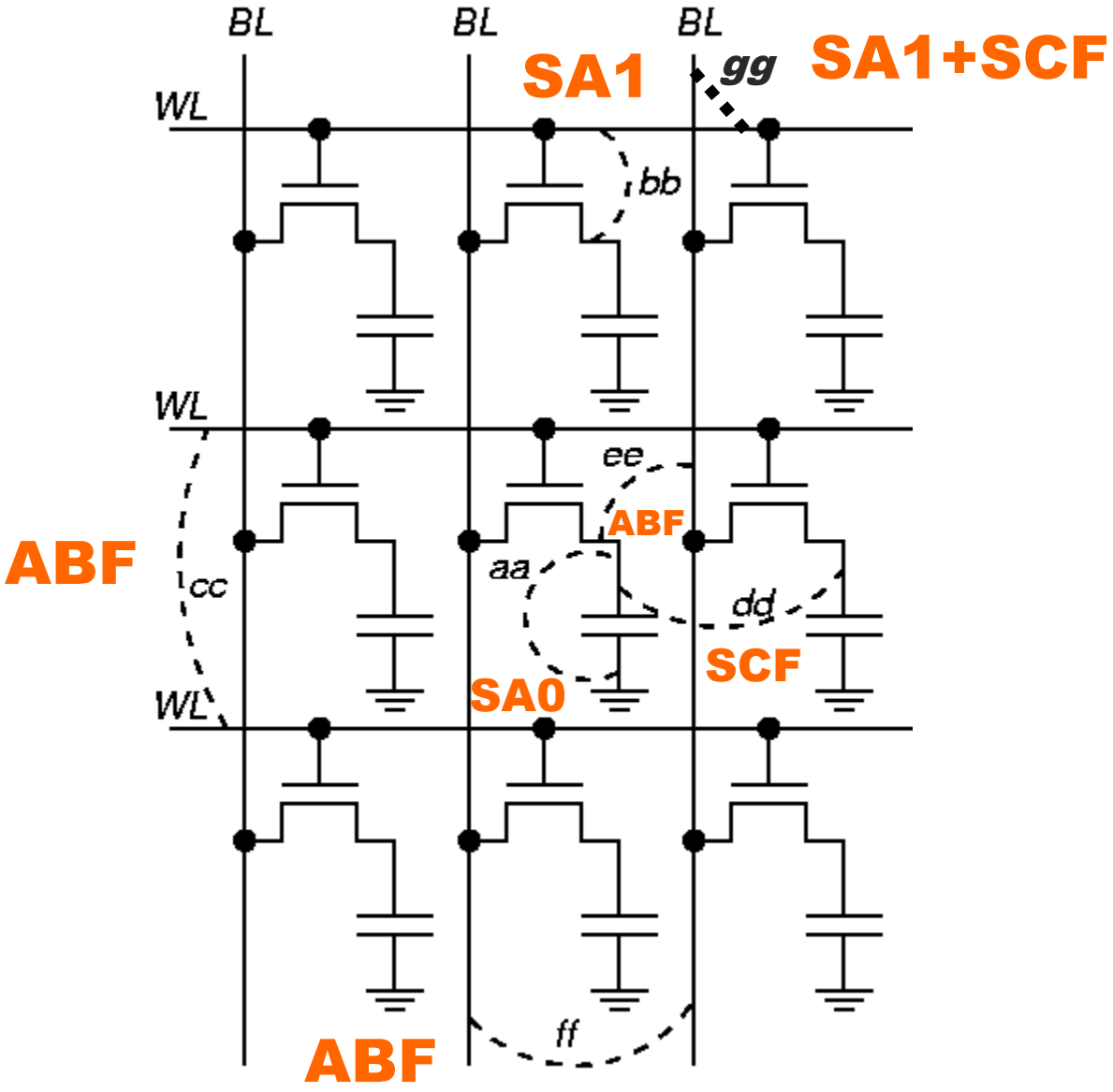


Examples of the Fault Models

- Internal defects (e.g. shorts, opens, bridges, etc.) often manifest themselves as one or a combination of those faults.

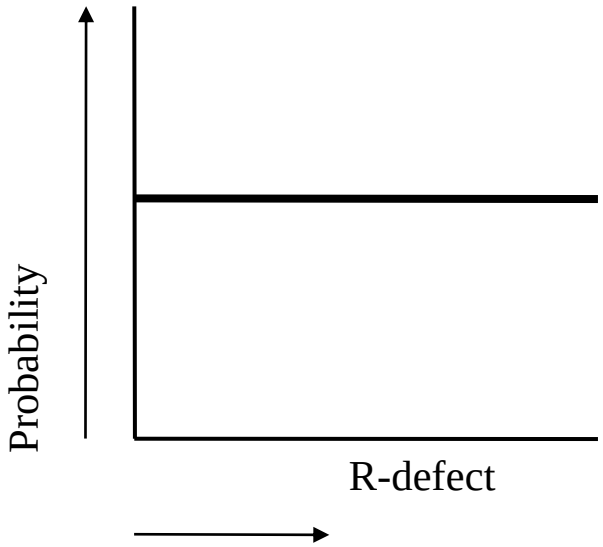


Examples of the Fault Models (cont.)



Validation of the Fault Models

- Performed by (Dekker, ITC'88) for SRAMs
- 8 K*8 = 64 Kbits, 4 Transistor (4T) cells
- technology: 4 μm (*Note*: R-defect is defect resistivity)
- Distribution for R-defect values assumed to be uniform
  - —i.e., no FAB statistics taken into account



Validation of the Fault Models (cont.)

- Table shows likelihood of functional faults, as a function of the spot defect size
- *Results:*
  Two new fault models established: SOpF and DRF
  - SAFs ≈ 50%
  - CFs due to large spot defects
  - **CFin not present**!

Functional fault class	Spot defect size (µm)	
	<2	<9
SAF	51.3%	49.8%
SOpF	21.0%	11.9%
TF	0%	7.0%
CFst	9.9%	13.2%
CFid	0%	3.3%
DRF	17.8%	14.8%
Total	100%	100%

Basic Concept of March Test

Notations for Describing Faults & March Tests

- 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
- ↕ -- Complement the cell contents
- ⇑ -- Increasing memory addressing
- ⇓ -- Decreasing memory addressing
- ⇳ -- Either increasing or decreasing

March Tests: Concept and Notation

- A *march test* consists of a sequence of *march elements*
- A *march element* consists of a sequence of operations applied to every cell, in either one of two *address orders*:
  1. *Increasing* (⇑) address order; from cell 0 to cell *n*-1
  2. *Decreasing* (⇓) address order; from cell *n*-1 to cell 0

  Note: The ⇑ address order may be *any sequence of addresses* (e.g., 5,2,0,1,3,4,6,7), provided that the ⇓ address order is the *exact reverse sequence* (i.e, 7,6,4,3,1,0,2,5)

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

MATS+ Test

- MATS+ {⇳(w0);⇑(r0,w1);⇓(r1,w0)}

  M0 M1 M2

  — Test consists of 3 march elements: M0, M1 and M2

  — The address order of M0 is irrelevant (Denoted by symbol ⇳)

  M0: **⇳(w0)** means:
  ```
  for i = 0 to n-1 do {
      A[i]:=0
  }
  ```
  M1: **⇑(r0,w1)** means:
  ```
  for i = 0 to n-1 do {
      read A[i];      //check if 0 is read
      A[i]:=1
  }
  ```
  M2: **⇓(r1,w0)** means:
  ```
  for i = n-1 to 0 do {
      read A[i];      //check if 1 is read
      A[i]:=0
  }
  ```

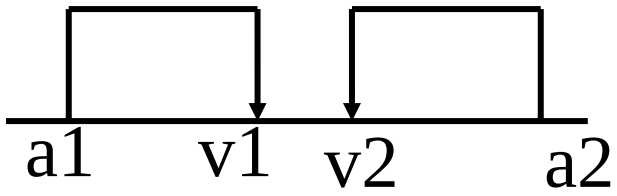
Combinations of Memory Cell Faults

A memory may contain:

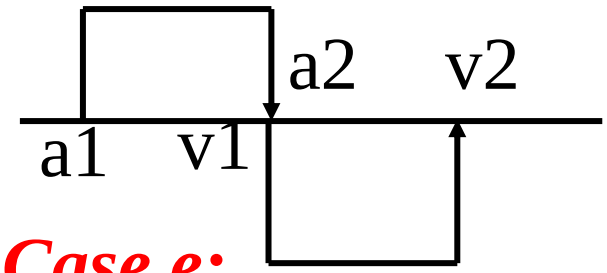
- A single fault
- Multiple faults (6 cases of CFs shown; ai is a-cell, vj is v-cell)) can be:
  - *Unlinked*: Faults do not interact (*Cases a, b, c, e*)
  - *Linked*: Faults do interact (***Cases d, f***)

    Linked faults have a *common victim*

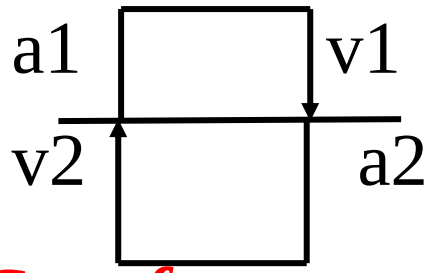
Case a:



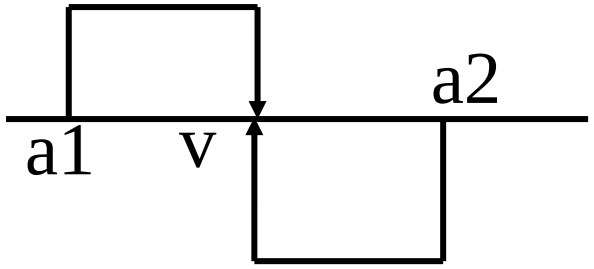
Case b:



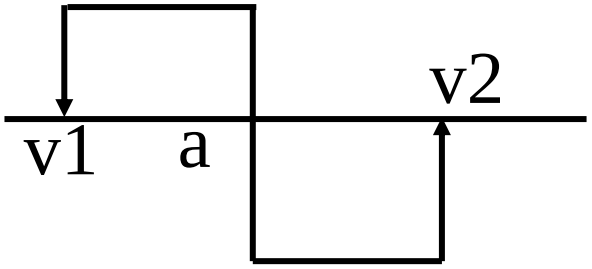
Case c:



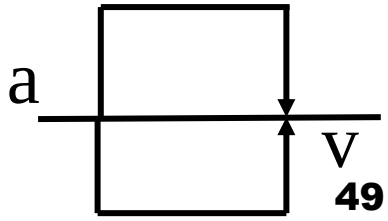
Case d:



Case e:



Case f:

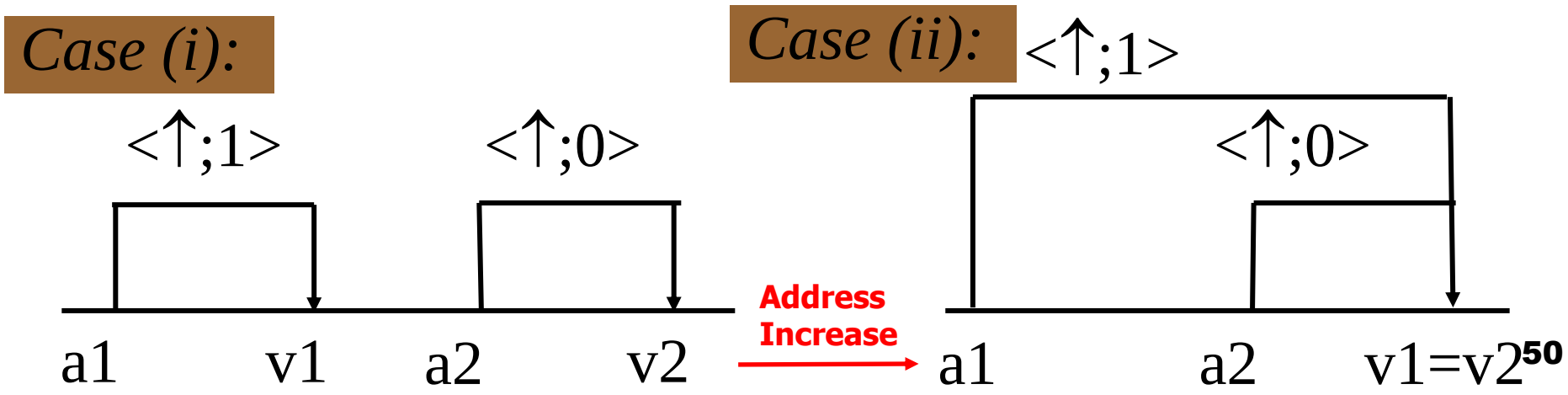


Linked Coupling Faults

- Case *(i)*: Two unlinked CFids; detected by the march test
  {⇳(w0); ⇑(r0,w1);⇓(w0,w1);⇓(r1)}
  M0 M1 M2 M3
  <↑;1>=<↑;↑> CFid *sensitized* by w1 operation of M1
  detected by r0 of M1
  <↑;0>=<↑;↓> CFid sensitized by w1 of M2, detected by r1 of M3

March elements

- Case *(ii)*: Linked CFids
  - Cannot be detected by test of Case *(i)* (for unlinked CFids) because of *masking*
  - Linked faults require special, more complex, tests



Address Decoder Faults (AFs)

- Functional faults in the address decoders:

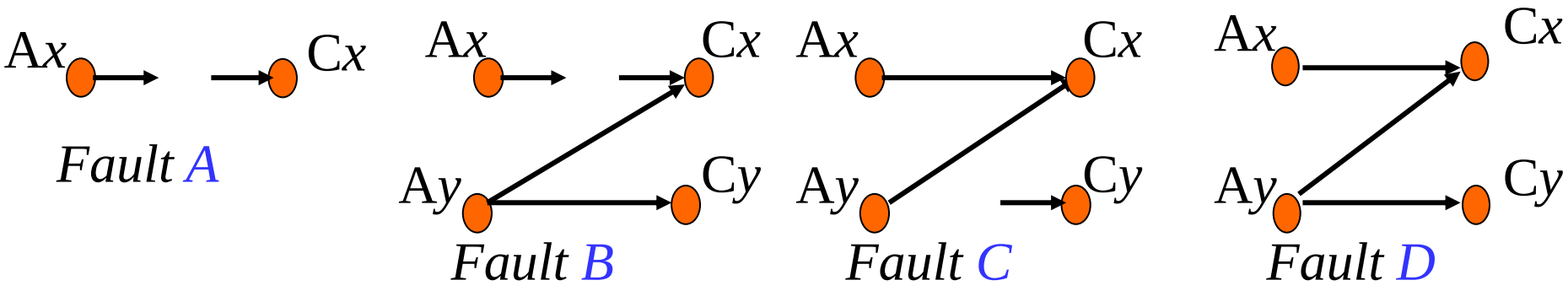
A. With a certain *address*, *no* cell will be accessed

B. A certain *address* accesses multiple cells

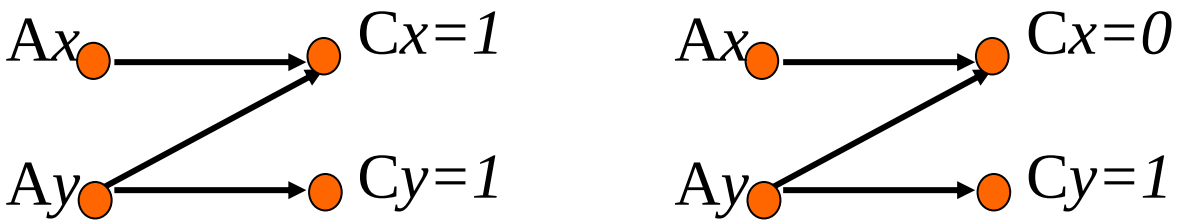
C. A certain *cell* is accessed by multiple addresses

D. Certain cells are accessed by their own and other addresses

Difficult fault: *Read operations may produce a **random** result*

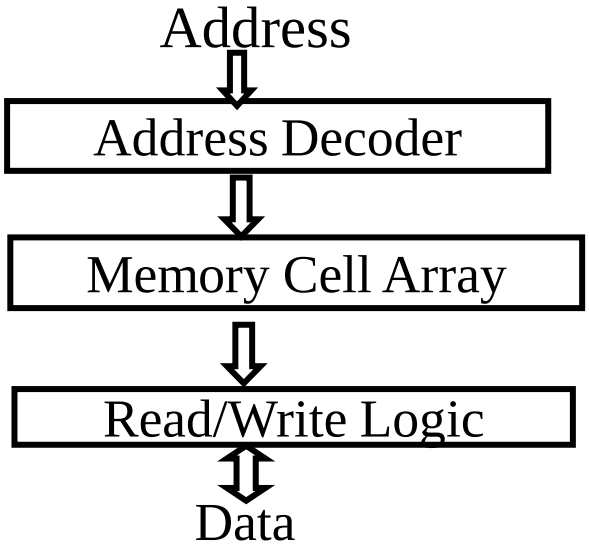


Fault D: Reading from address Ay

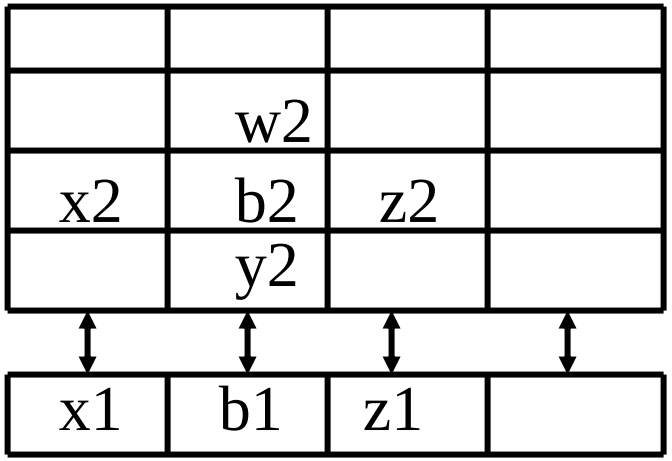


Mapping Read/Write Logic Faults

- The reduced functional model consists of *three* blocks
  1. the address decoder
  2. the memory cell array (MCA)
  3. the read/write logic



- Read/write logic faults can be mapped onto MCA faults
  - SAFs, TFs and CFs will be detected by tests for the MCA



Mapping Address Decoder Faults

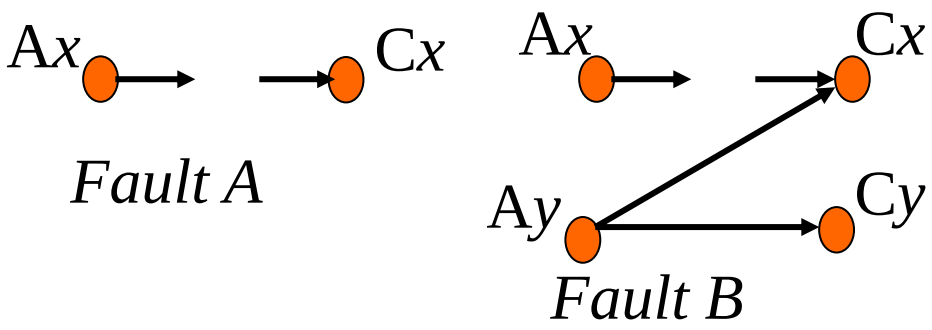
- March tests for MCA faults detect AFs if they satisfy condition:

Cond. AF: The march test has to contain the following march elements

1. ⇑(r*x*,…,w*x**) This means either ⇑(r0,…,w1) or ⇑(r1,…,w0)
2. ⇓(r*x**,…,w*x*) Note: '…' means any # of *r* or *w* operations

Proof: (Easy for **Faults A, B**)

- AFs A and B will be detected by every test which detects SAFs in the memory cell array. When address Ax is written into and then read from, depending on the technology, cell Cx will appear to be SA0 or SA1. Thus, either condition 1 or 2 (depending on whether x is 0 or 1 in the test) will detect the fault.



Mapping Address Decoder Faults (cont.)

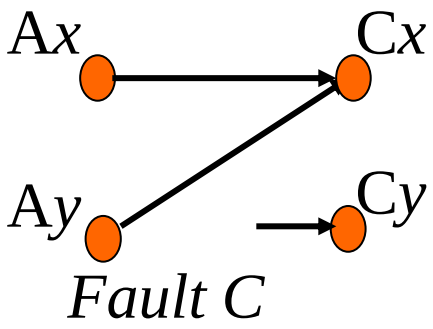
- March tests for MCA faults detect AFs if they satisfy condition:

Cond. AF: The march test has to contain the following march elements

1. ⇑(r*x*,…,w*x**) This means either ⇑(r0,…,w1) or ⇑(r1,…,w0)
2. ⇓(r*x**,…,w*x*) Note: '…' means any # of *r* or *w* operations

Proof: (Still easy for **Fault C**)

- AF C is detected by first initializing the entire memory array to a certain value (the expressed value h which can be x or x'). Thereafter, any march element which reads the expected value h and ends with writing the cells with h', will detect fault C. Thus, any one of the conditions 1 (where x=h) or 2 (where x'=h) will detect fault C.



Mapping Address Decoder Faults (cont.)

- March tests for MCA faults detect AFs if they satisfy condition:

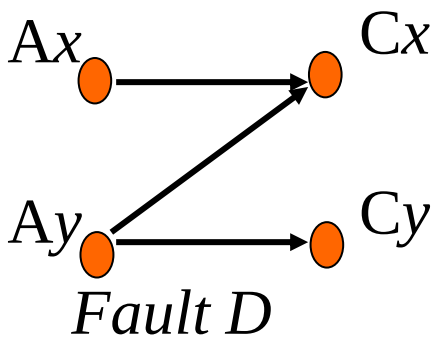
Cond. AF: The march test has to contain the following march elements

1. ⇑(r*x*,…,w*x**) This means either ⇑(r0,…,w1) or ⇑(r1,…,w0)
2. ⇓(r*x**,…,w*x*) Note: '…' means any # of *r* or *w* operations

Proof:

- For **Fault D** the result of a read operation can be:
  - A deterministic function (Logical AND or OR) of read values
  - A *random result* (when the read cells contain different values)

(See next page for details of proof).



Mapping Address Decoder Faults (cont.)

- Fault D: read result *random* if cells contain *different values*
- Cond. AF-1: ⇑(r*x*,…,w*x**) detects Faults D1 & D2
  - When A*x* written with *x**, cells C*y*…C*z* are also written with *x**
  - Fault detected when C*y* is read: reads *x** while expecting *x*
- Cond. AF-2: ⇓(r*x**,…,w*x*) detects Faults D1 & D3
  - When A*x* written with *x*, cells C*v*…C*w* are also written with *x*
  - Fault detected when C*w* is read: reads *x* while expecting *x**