# Chapter 7 Memory Testing

Jin-Fu Li
Advanced Reliable Systems (ARES) Laboratory
Department of Electrical Engineering
National Central University
Jhongli, Taiwan

# Outline

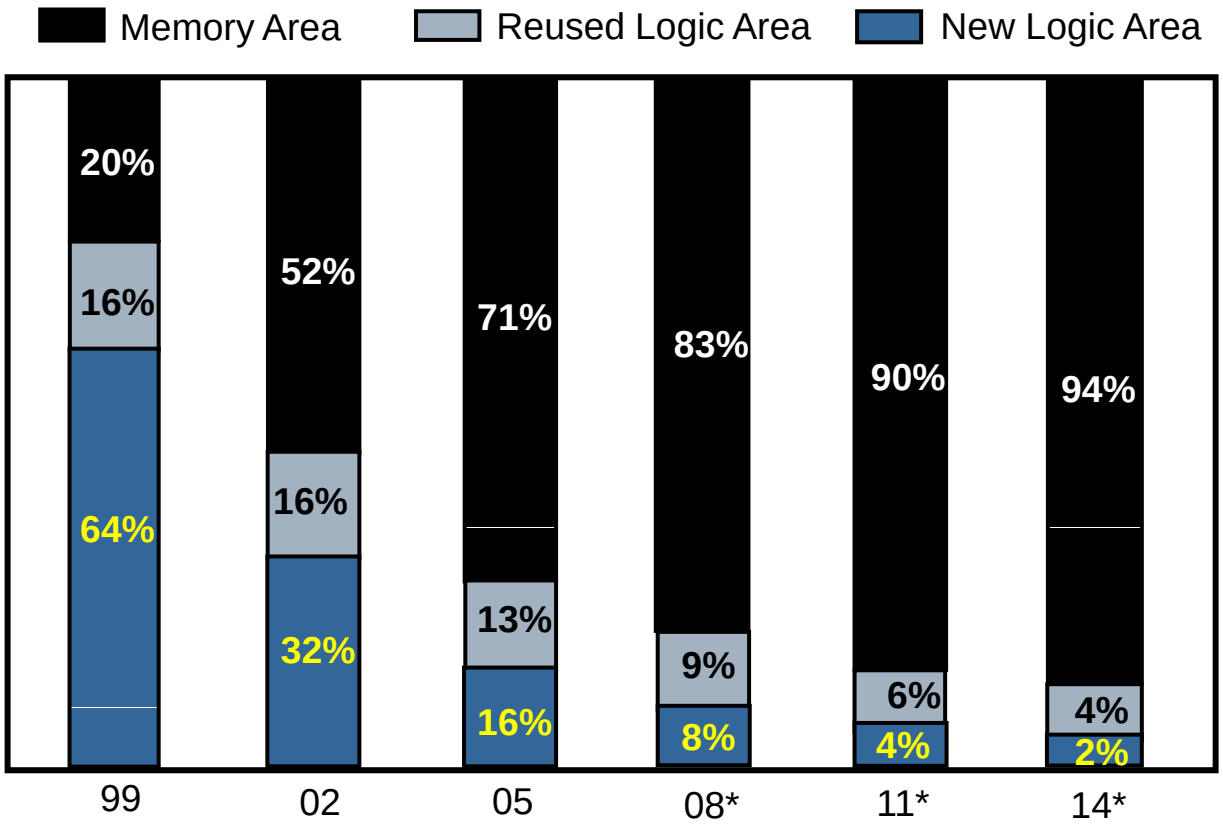
- Importance of Embedded Memories
- RAM Functional Faults
- March Tests
- Converting Bit-Oriented RAM Tests into Word-Oriented RAM Tests
- RAM BIST

# Embedded Memory–The Key to SOC

- Embedded memory is becoming more central to integrated circuit design
- Historically, ICs were dominated by the logic functions, with memory being external
- Today, an SOC contains many memory blocks of different sizes, shapes and functionality
  - Typically, embedded memories represent about 30%~50% SOC area
- The Semiconductor Industry Association (SIA) predicts that 90% of the SOC's surface will be memory by 2011

# Embedded Memory–The Key to SOC

- SOC memory continues to increase




Source: SIA, ITRS, 2000

# Embedded Memory–Quality

- During manufacture
  - Yield
  - Exponential yield model
  - $Y = e^{-\sqrt{AD}}$, where $A$ and $D$ denote the area and defect density, respectively
- After manufacture
  - Reliability
- During use
  - Soft error rate

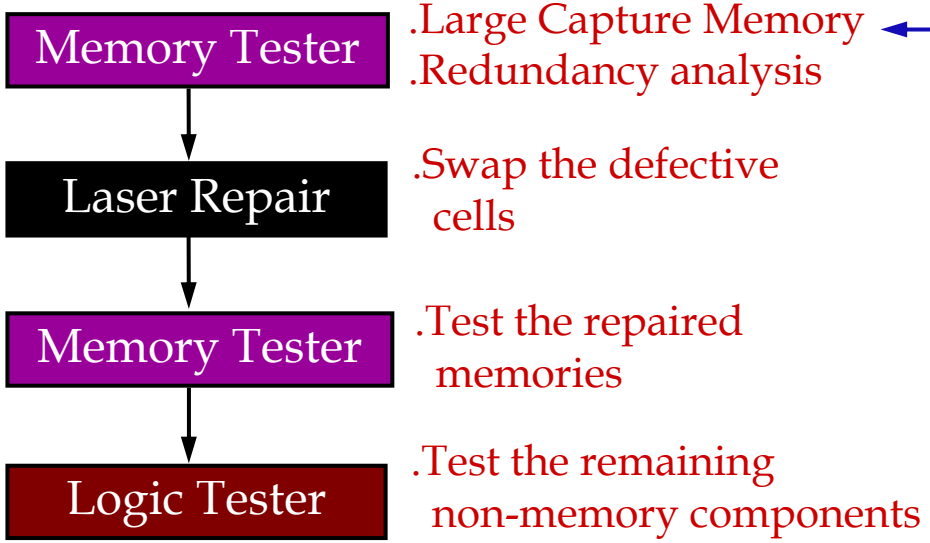
# Quality During Manufacture

- Issues of embedded memory
  - As the density of transistors is increased, the D is increased compared to logic
  - About 2X logic for high density 6-T SRAM
- Solutions
  - Redundancy & laser repair using ATE
  - Error correction code (ECC)
- Redundancy & repair
  - Achieve yield parity with logic, or better
  - About 3% area overhead
  - Recommended over 1Mb
  - Laser repair manufacturing flow

# Quality During Manufacture

- ECC
  - Detect/repair defects in individual words
  - 25-30% area overhead
    - Ex: 6 extra bits for single bit correction in 32 bit words
  - Latency penalty
    - At least one clock cycle latency penalty

# Quality-Insurance Methods for Memories in SOCs




.Large Capture Memory
.Redundancy analysis

.Swap the defective cells

.Test the repaired memories

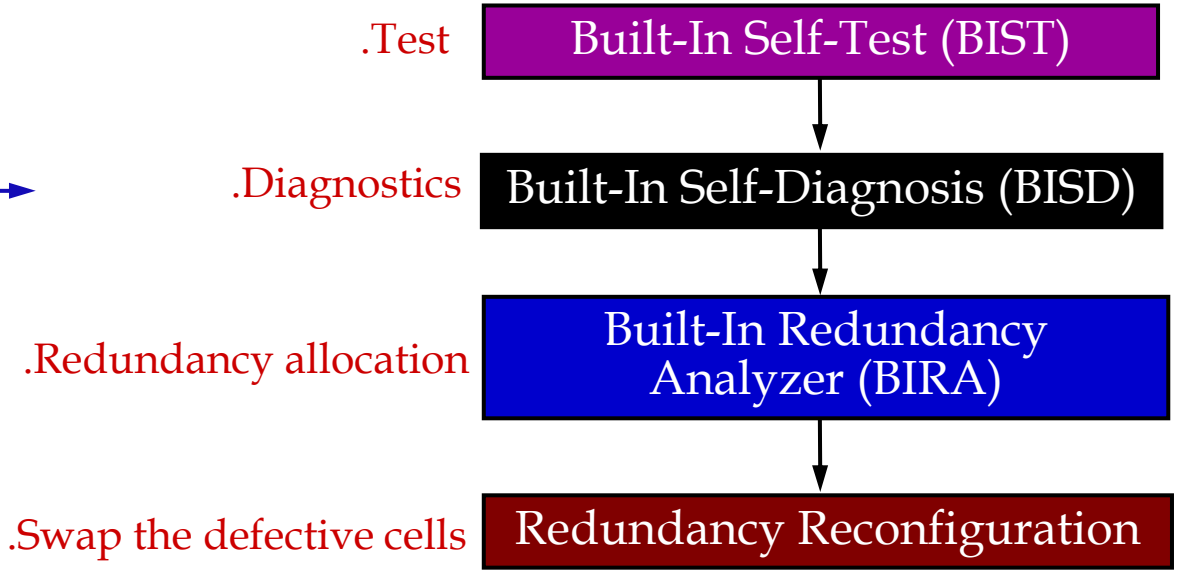
.Test the remaining non-memory components

Conventional test and repair approaches

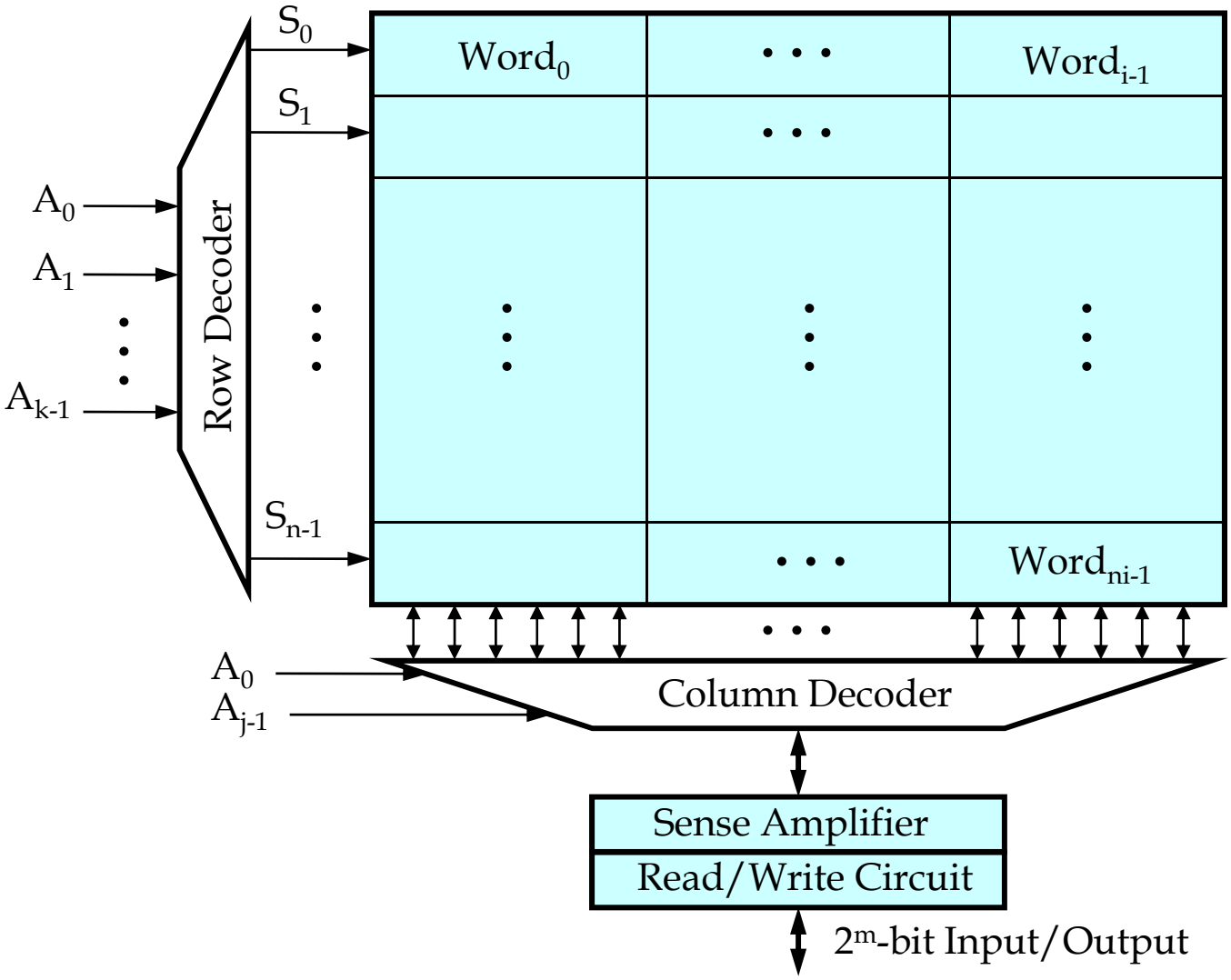
1. Long testing time
2. Expensive memory tester and logic tester are needed

BIST & BISR approaches

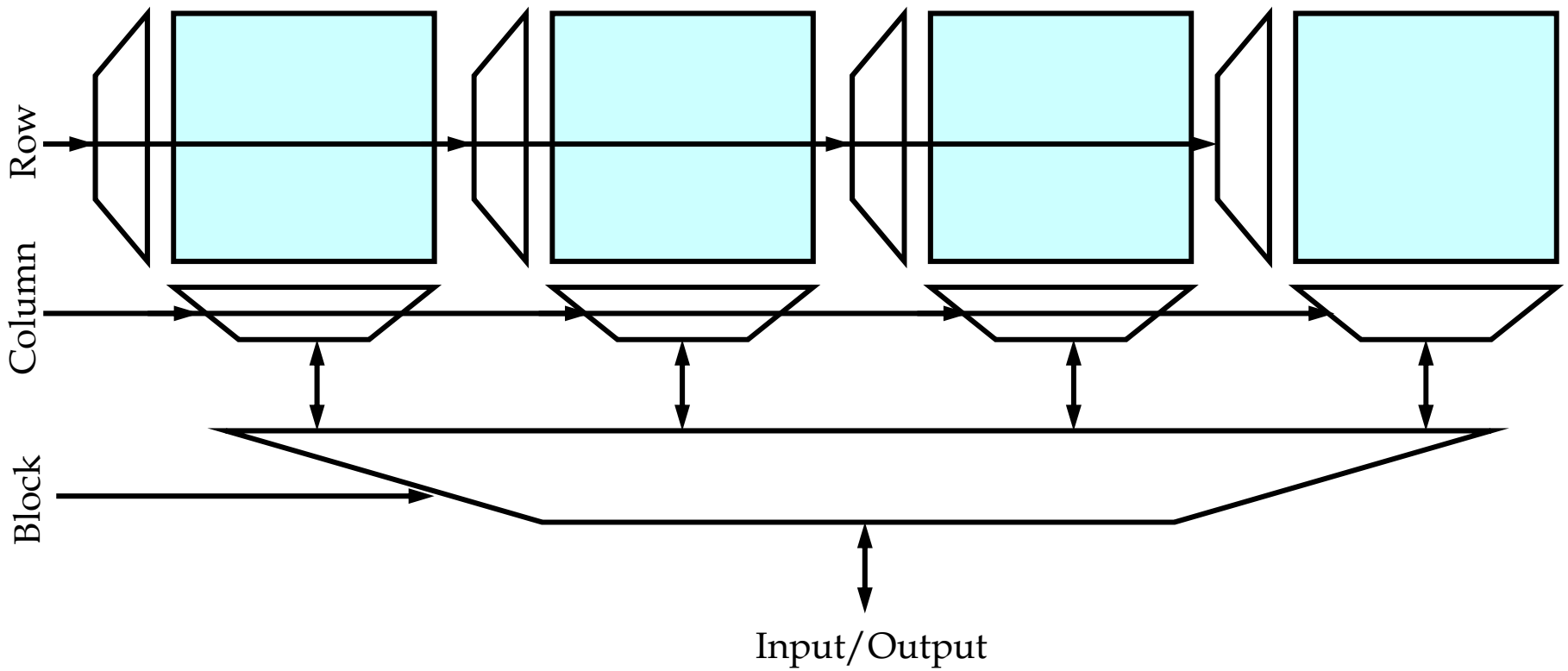
1. Short testing time
2. Only cheap logic tester is needed



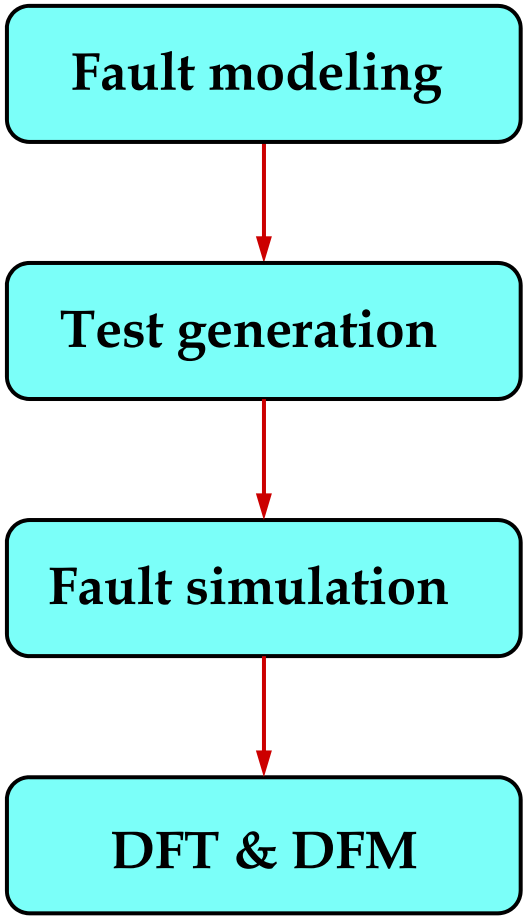
# 2-D Memory Architecture



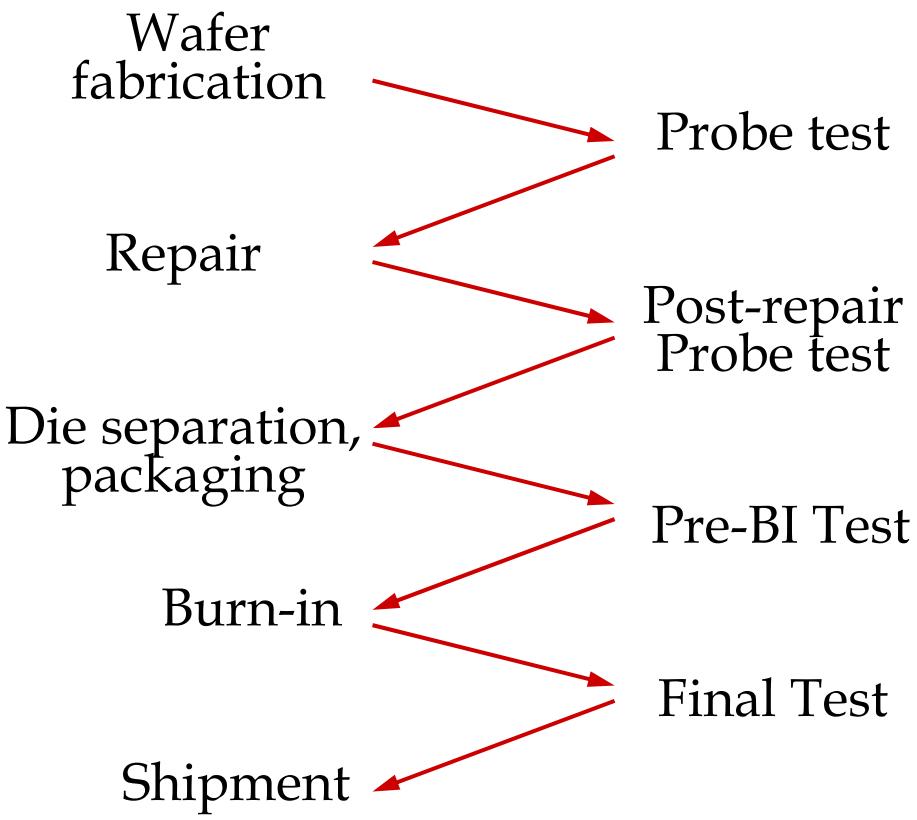
# 3-D Memory Architecture



# Memory Testing

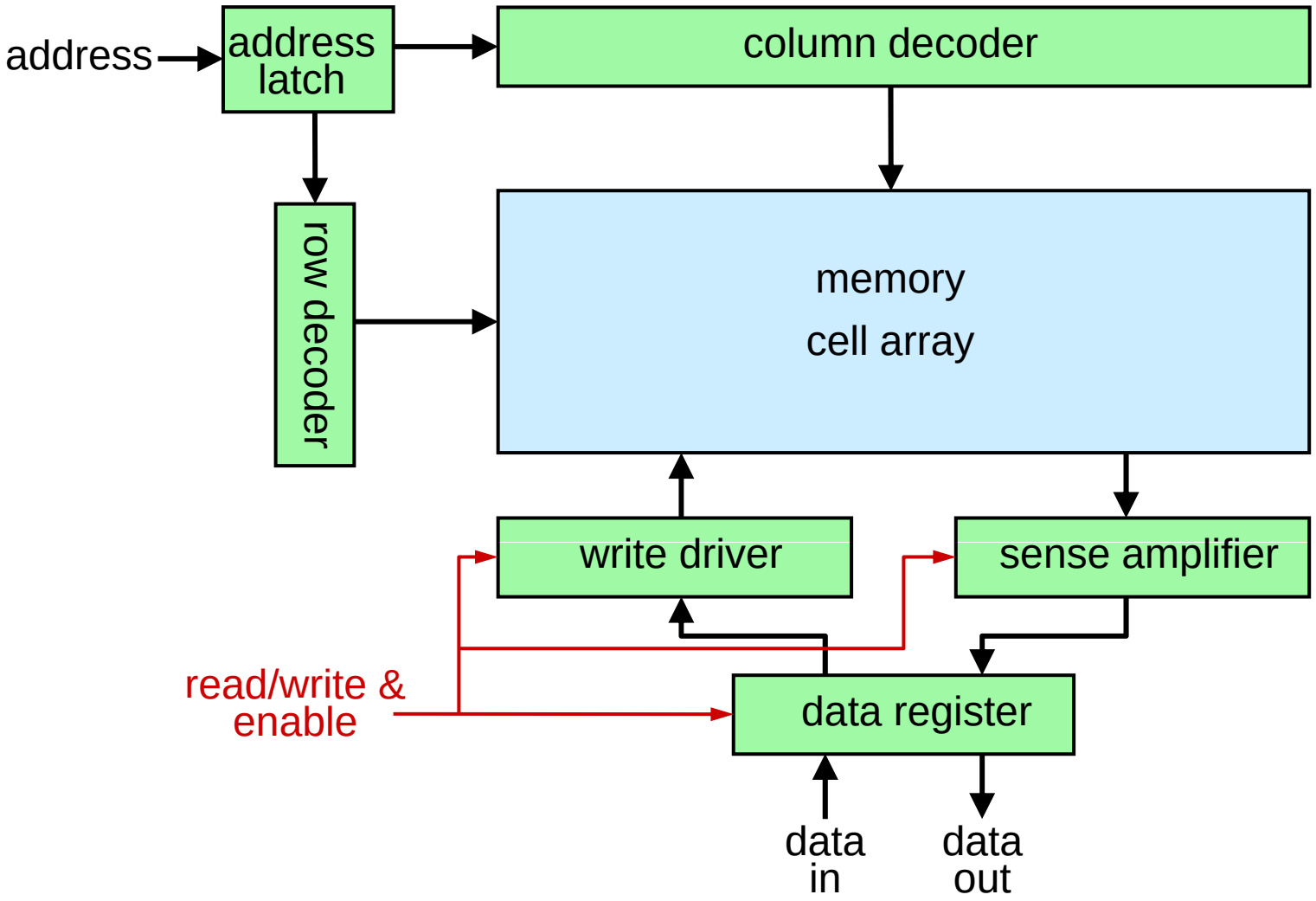



Test-Preparation Phase

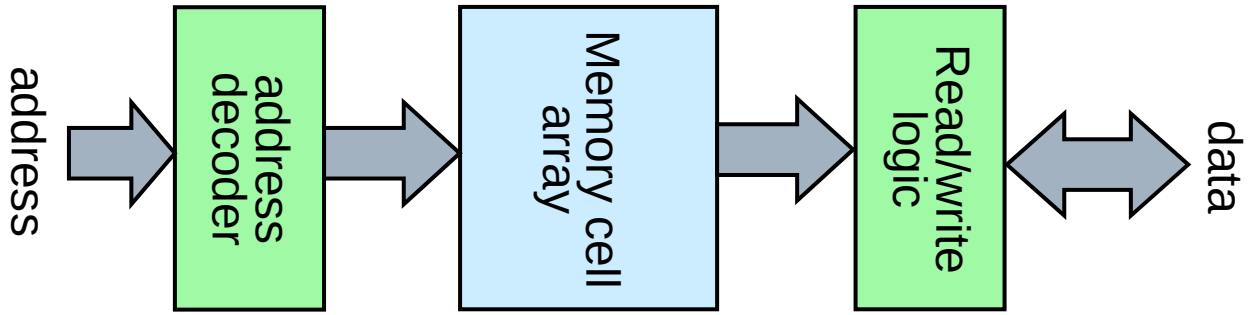



Test-Execution Phase

# Functional RAM Model



# Reduced Functional Model



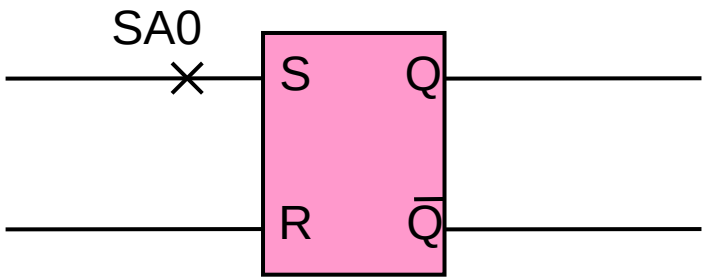

- The address latch, the row, and the column-decoder are combined to the address decoder
  - They all concern addressing the right cell or word
- The write driver, the sense amplifier, and the data register are combined to the read/write logic
  - They all concern the transport of data from and to the memory cell array

# Reduced Functional Faults

- Stuck-at fault (SAF)
  - Definition:
    - The logic value of a stuck-at (SA) cell or line is always 0 or 1.
    - It is always in state 0 or in state 1 and cannot be changed to the opposite state
  - Detection requirement:
    - From each cell or line, a 0 or 1 must be read
- Transition fault (TF)
  - Definition:
    - A cell that fails to undergo a *0* to *1* transition when it is written is said to contain an *up transition fault*
    - A *down transition fault* indicates that a cell fails to undergo a *1* to *0* transition

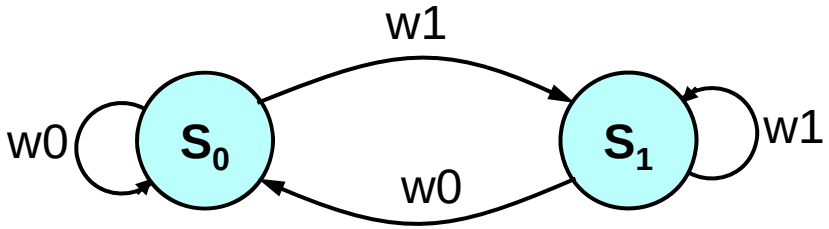
# Reduced Functional Faults

- A TF can be thought of as a set/reset (S/R)-type flip-flop with a SAF on the set or reset input

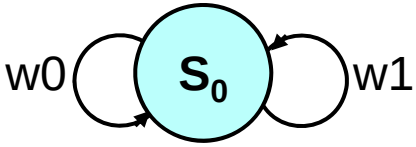



- Detection requirement:
  - Each cell should undergo up and down transitions and be read after each transition before undergoing further transitions

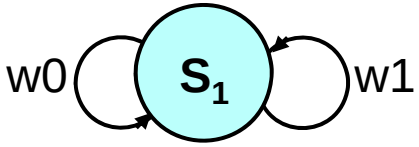
# State Diagram for SAFs & TFs



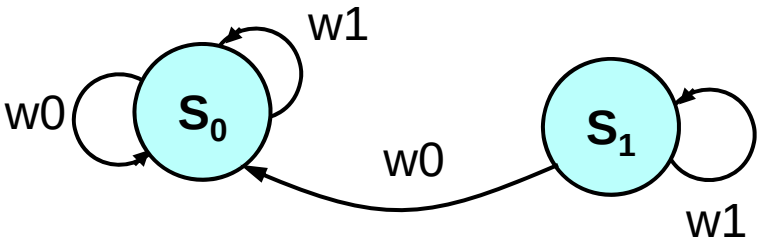

State diagram of a good cell




SA0 fault




SA1 fault

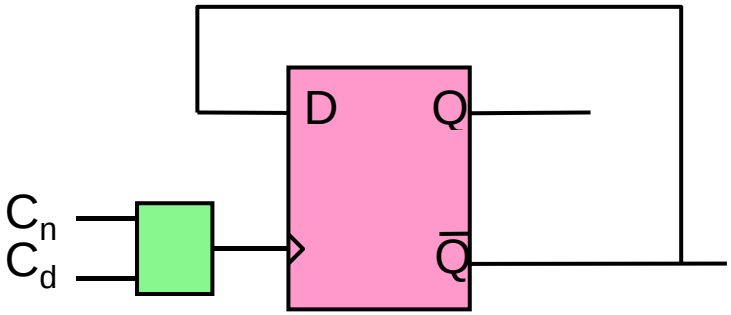



TFu

# Reduced Functional Faults

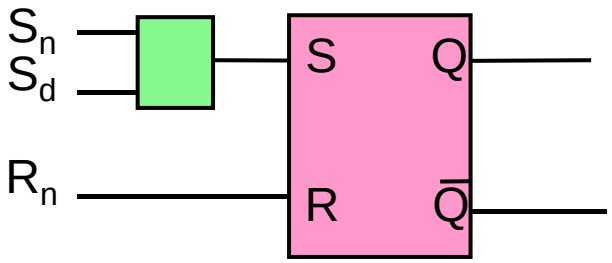
- Coupling faults (CFs)
  - 2-cell coupling faults
  - Different types of CFs
    - Inversion CF (CFin)
    - Idempotent CF (CFid)
    - State CF (CFst)
    - Dynamic CF (CFd)
- CFin
  - A 0 to 1 (or 1 to 0) transition in one cell inverts the content of a second cell
  - An CFin can be thought of as a D-type flip-flop with an extra clock input $C_d$ and the Q' output tied to the D input, as depicted in the following figure.

# Reduced Functional Faults



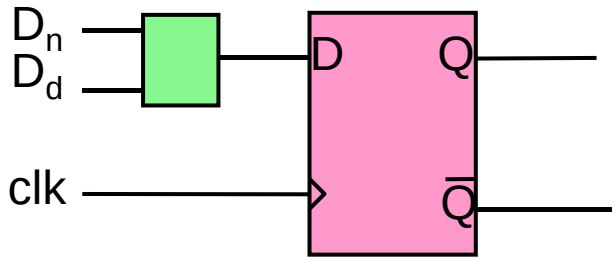

- CFid
  - A 0 to 1 (or 1 to 0) transition in one cell forces the content of a second cell to a certain value, 0 or 1.
  - An idempotent coupling fault can be thought of as an S/R-type flip-flop with an OR-gate in the Set or Reset line, as depicted in the following figure. $S_n$ is the normal set input whereas $S_d$ is the undesired set input due to coupling with one or more other flip flops.

# Reduced Functional Faults



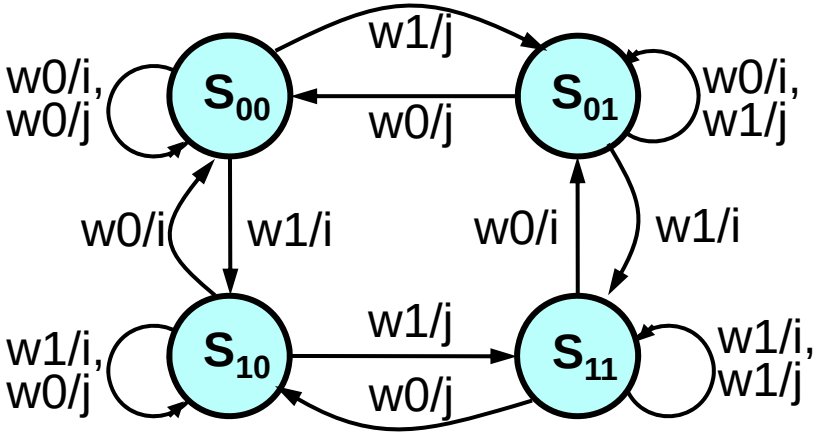

- CFst
  - A 0 or 1 state in one cell forces the content of a second cell to a certain value, 0 or 1
  - It can be thought of as a D-type flip-flop with an OR/AND-gate in the data line (D), as depicted in the following figure. $D_n$ is the normal set input whereas $D_d$ is the undesired set input due to coupling with one or more other flip flops

# Reduced Functional Faults

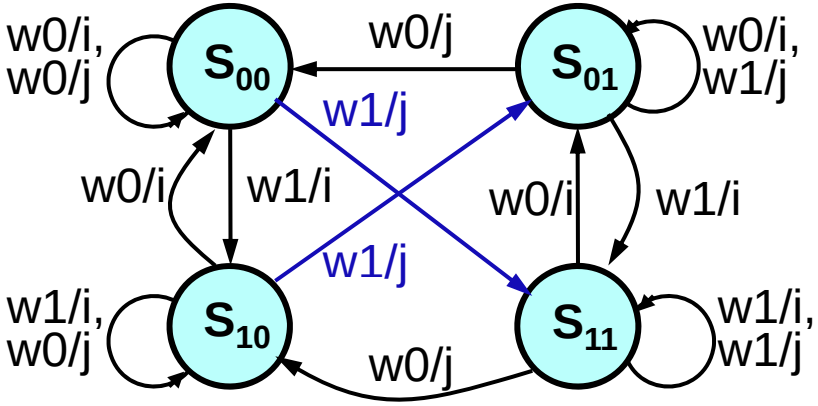



- Dynamic Coupling Fault ($CF_d$)
  - Occur between cells in different words. A *read* or *write* operation on one cell forces the content of the second cell either 0 or 1.
  - Denoted as <rx|wy; z> where | denotes the *or* of the read and write operations
  - Four possible $CF_d$ faults
    - <r0|w0;0>, <r0|w0;1>,<r1|w1;0>, and <r1|w1;1>

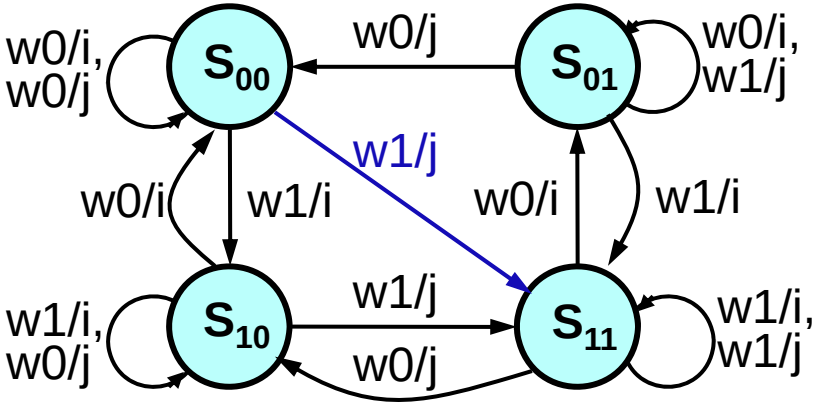
# State Diagrams for CFin & CFid




State diagram of two good cells




State diagram of an CFin<u;i>

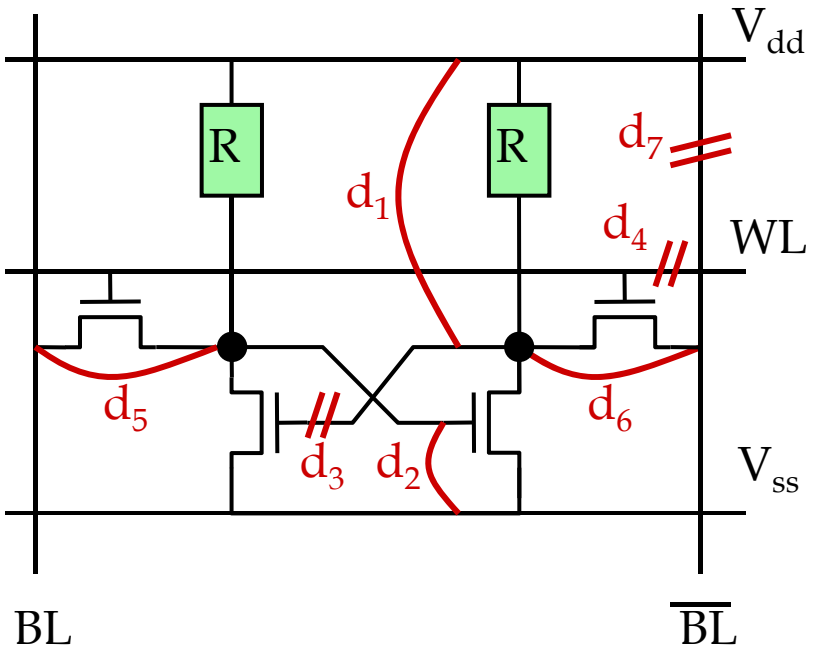



State diagram of an CFid<u;1>

# Summary of CFs

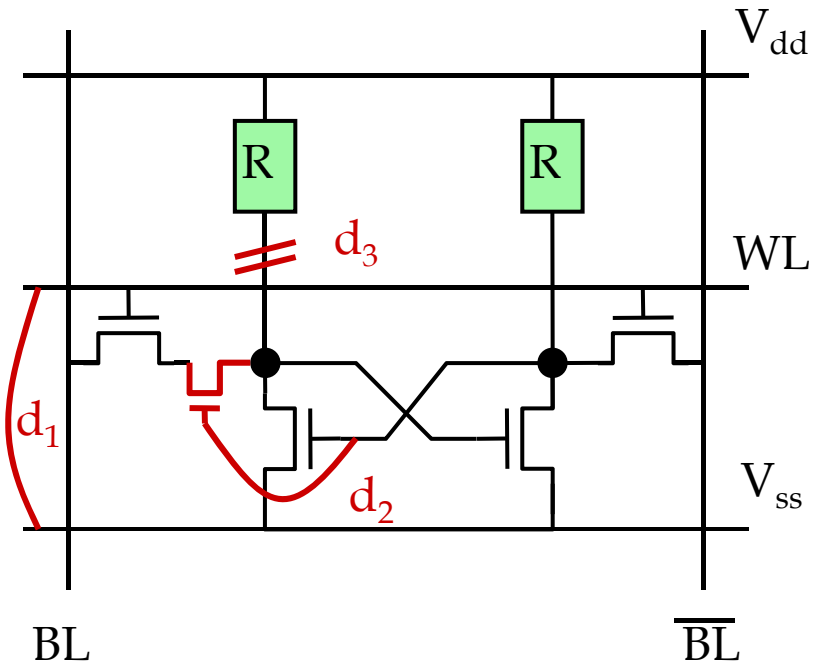
- Note that all definitions talk about single-way faults, that is, the presence of a CF from *cell i* to *cell j* does not imply the presence of a CF from *cell j* to *cell i*.
- Suppose that a transition or state in cell j can induce a coupling fault in cell i. Cell i is then said to be *coupled* cell (or *victim*); cell j is called the *coupling* cell (or *aggressor*).
- A test that has to detect and locate all coupling faults should satisfy
  - For all coupled cells, each cell should be read after a series of possible coupling faults may have occurred

# Relation Between Faults & Defects




- Defect $d_1$ (inverse node shorted to $V_{dd}$) causes a SA0 fault
- Defect $d_2$ (true node shorted to $V_{ss}$) causes a SA0 fault
- Defect $d_3$ (open true node gate) cause a SA1 fault
- Defect $d_4$ (an open word line) causes all cells after the WL fault to be inaccessible (AF)
- Defect $d_5$ (a short between the true node and BL) will pull BL down if the cell contains a 0, but will not affect BL if the cell contains 1. This is the state coupling fault <0;0>
- Defect $d_6$ (short between inverse node and BL') is similar, as is the state coupling fault <1;1>
- Defect $d_7$ (open BL') prevents cells after the open defect from passing a logic 1 value on BL'

# Relation Between Faults & Defects




- Defect $d_1$ (a word line connected to $V_{ss}$) causes all cells in the word line to be *stuck open*.
- A defect in the poly silicon layer covering a diffusion region may result in the creation of an extra pass transistor. This defect ($d_2$) causes a *transition fault*.
- Defect $d_3$ (a broken pull up resistor) introduces a *data retention fault*. If the cell is not accessed, the cell node with the broken pull up resistor can be floating high or active low. If the node is floating high, the leakage current from the cell node to the substrate will decline the voltage at the node. If the node voltage passes the threshold voltage $V_{th}$ the data in the cell will invert. If the node is active low, the cell will function correctly.

# March Tests

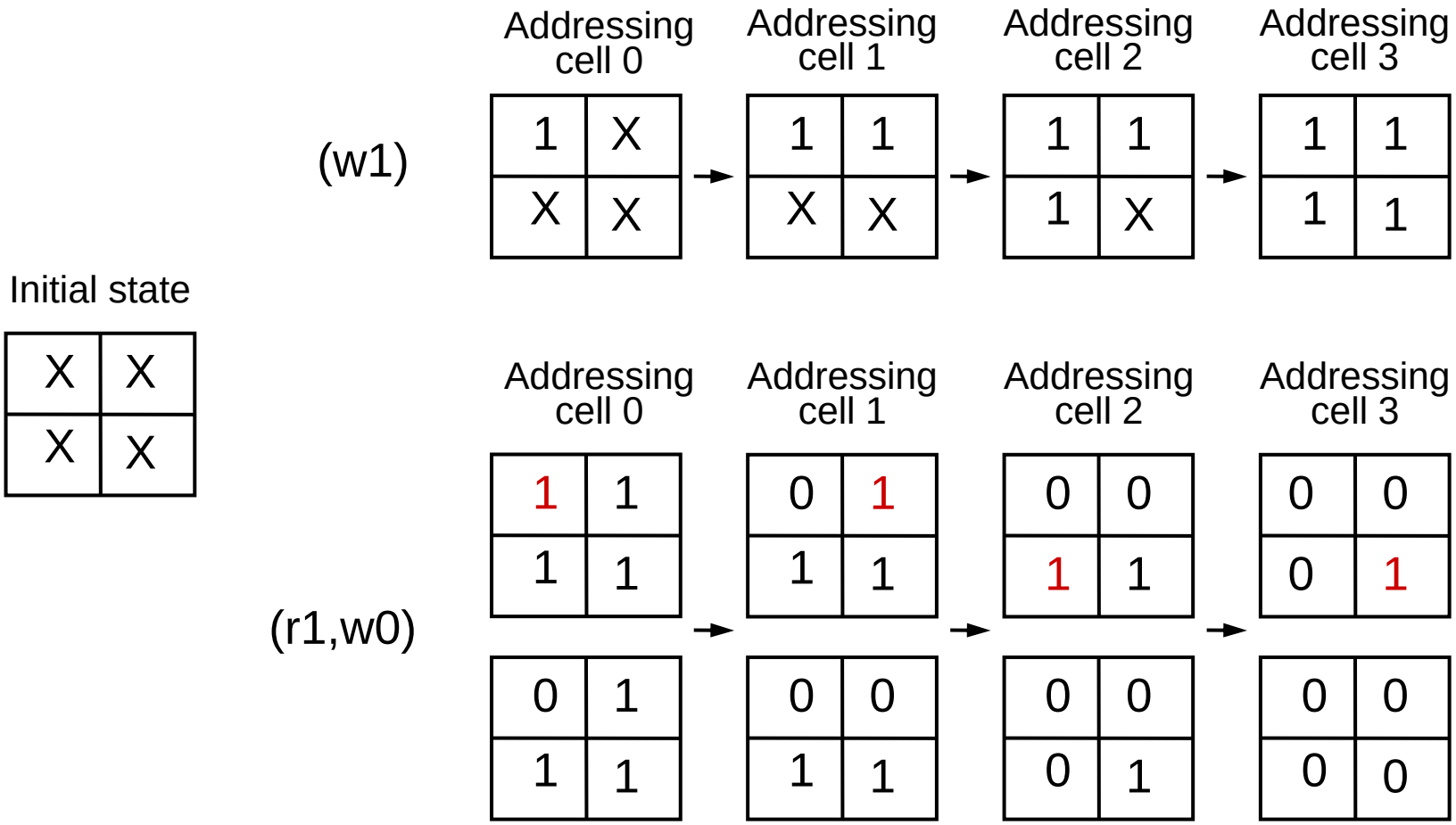
- A *march test* consists of a finite sequence of *march elements*
- A march element
  - A finite sequence of Read and/or Write operations applied to every cell in memory in either increasing address order (cell 0 to cell n-1) or decreasing address order (cell n-1 to cell 0)
- All operations of a march element are done before proceeding to the next address
- The march tests are a preferred method for RAM testing
  - Linear complexity, regularity, and symmetry

# March Test Notation

- r$x$: a read $x$ operation
- w$x$: a write $x$ operation
- $\Uparrow$: increasing addressing sequence (from 0 to n-1)
- $\Downarrow$ : decreasing addressing sequence (from n-1 to 0)
- $\Updownarrow$ : either increasing or decreasing addressing sequence

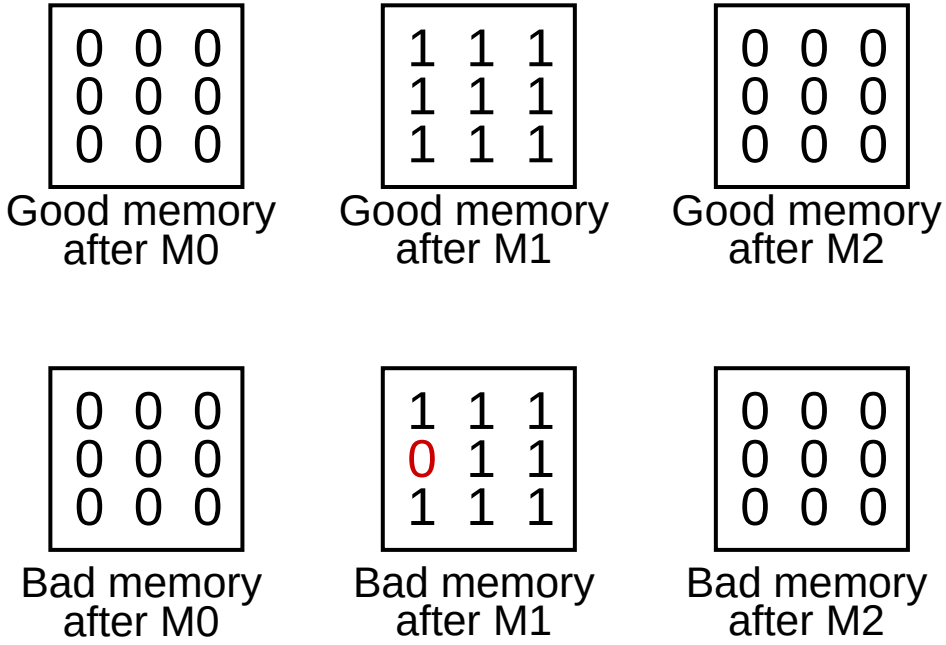
# An Example of March Test

- An example of march test $\{\Uparrow (w1); \Uparrow (r1, w0)\}$



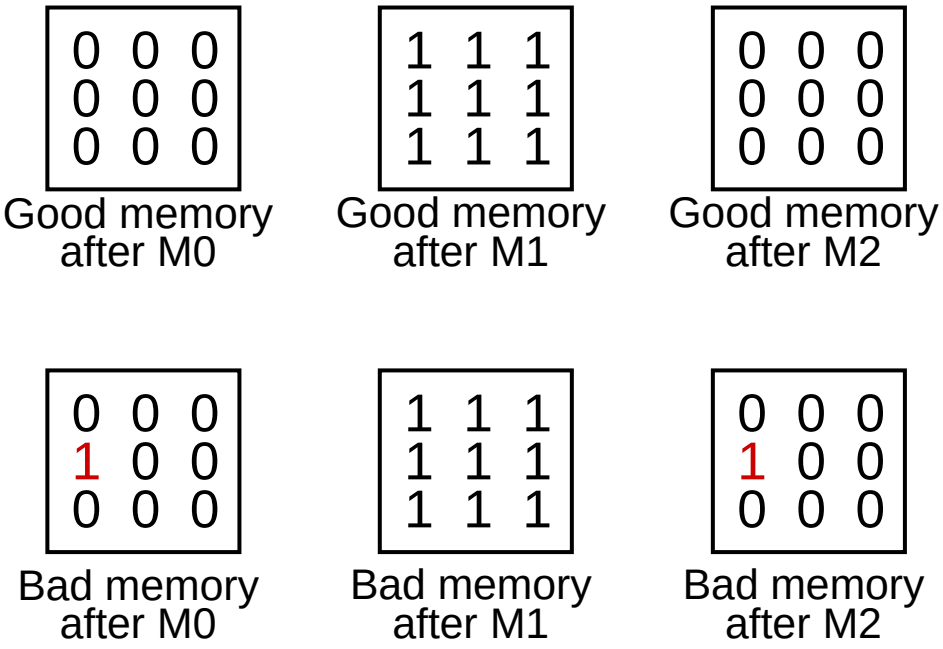
# March Tests for SAFs & TFs

- MATS+: $\{\Updownarrow(w0);\Uparrow(r0,w1);\Downarrow(r1,w0)\}$
- MATS+ detection of SA0 fault



# March Tests for SAFs & TFs

- MATS+ detection of SA1 fault



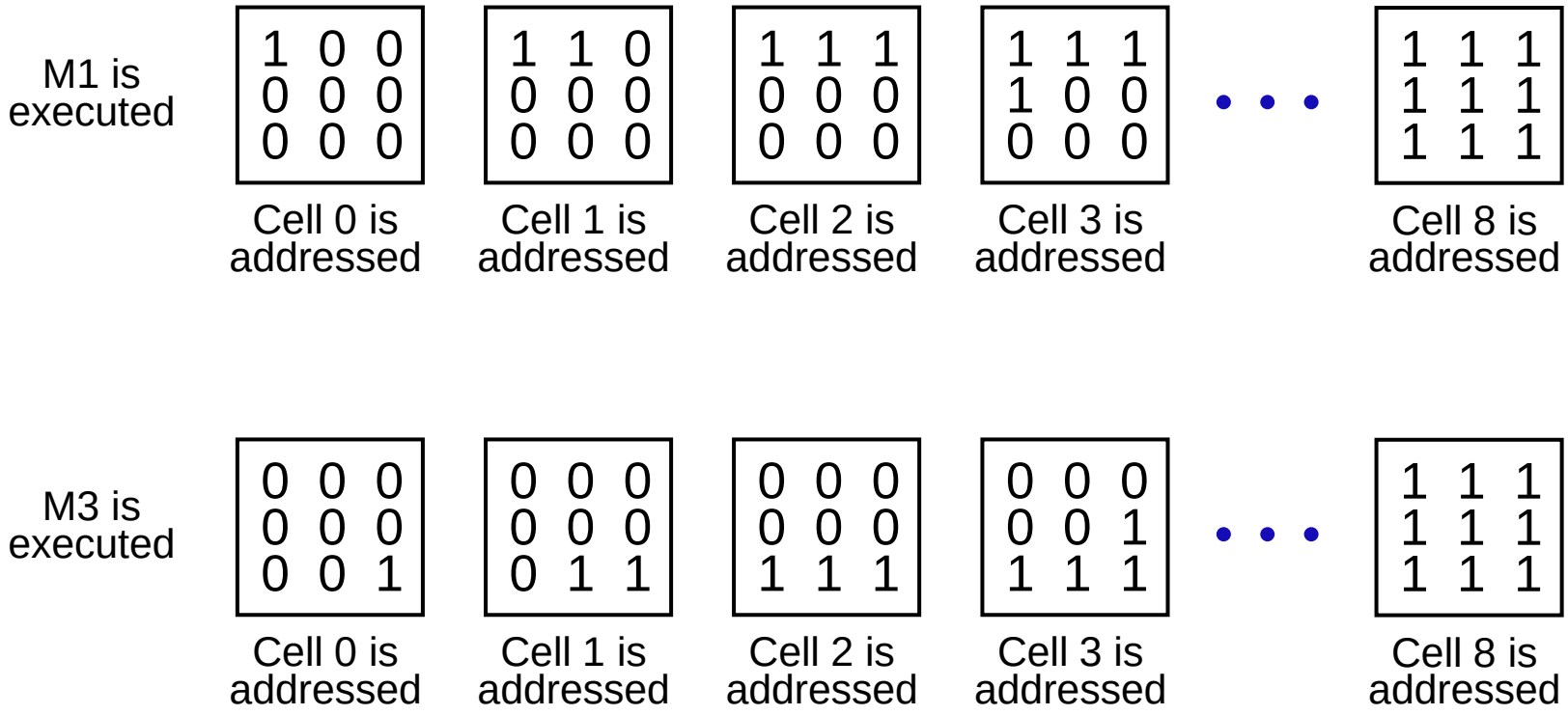

- MATS+ detection of TFu & TFd can be proved in the same way

# Tests for Detecting SAFs & TFs

- Conditions for detecting SAFs & TFs
  - SAFs & TFs can be detected by a march test which contains the following two march elements (or single march element containing both elements)
- $(\ldots, w0, r0, \ldots)$ to detect SA1 faults and TFd
- $(\ldots, w1, r1, \ldots)$ to detect SA0 faults and TFu

# March Tests for CFs

- March C -: $\{\Updownarrow(w0);\Uparrow(r0,w1);\Uparrow(r1,w0);\Downarrow(r0,w1);\Downarrow(r1,w0);\Updownarrow(r0)\}$
- Detection of CFs



# March Tests for CFs

- Conditions for detecting CFs
  - A march test which contains one of the two pairs of march elements of Case A & Case B can detect simple CFs (CFin, CFst, CFid)
  - Case A
    - 1. $\Uparrow(rx,\cdots,w\bar{x})\ \Uparrow(r\bar{x},\cdots,wx)$
    - 2. $\Downarrow(rx,\cdots,w\bar{x})\ \Downarrow(r\bar{x},\cdots,wx)$
  - Case B
    - 1. $\Uparrow(r\bar{x},\cdots,wx)\ \Uparrow(rx,\cdots,w\bar{x})$
    - 2. $\Downarrow(r\bar{x},\cdots,wx)\ \Downarrow(rx,\cdots,w\bar{x})$
- A.1 (A.2) will sensitize the CFs, and it will detect the fault, when the value of the fault effect is x′ (x), by the rx (rx′) operation of the first (second) march element when the coupled cell has a higher (lower) address than the coupling cell

# March Tests for DRFs

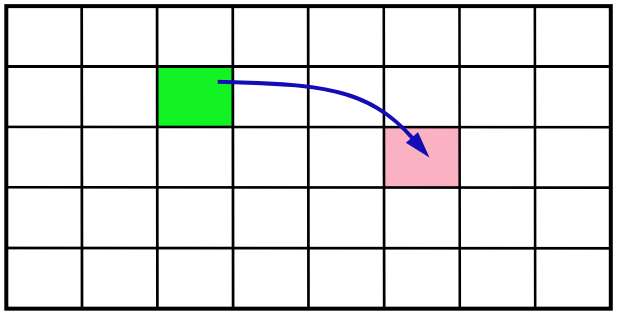
- Data retention faults (DRFs)
  - DRF has two subtypes
    - A stored '1' will become a '0' after a time T
    - A stored '0' will become a '1' after a time T
- Conditions for detecting DRFs
  - Any march test can be extended to detect DRFs
  - The detection of each of the two DRF subtypes requires that a memory cell be written into the corresponding logic states
- If we are interested in detecting simple DRFs only
  - The delay elements can be placed between any two pairs of march elements, e.g., $\Uparrow(rx,\cdots,w\bar{x})$; Del; $\Downarrow(r\bar{x},\cdots,wx)$

# Tests for Word-Oriented Memories

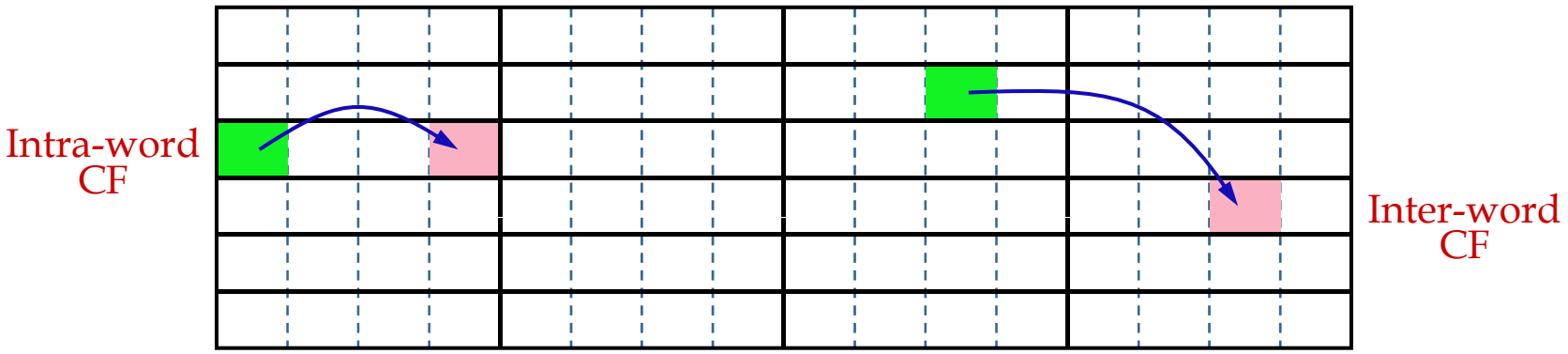
- Fault models for word-oriented memories (WOMs)
  - Only the class of memory cell array faults for bit-oriented memories (BOMs) has to be extended in order to cover WOMs
- The fault models for WOMs can be classified into two classes
  - Single-cell faults
    - SAFs, TFs, data retention faults (DRFs), etc.
  - Faults between memory cells
    - CFs
- Two classes of faults between memory cells for WOMs needed to be considered

# Tests for Word-Oriented Memories

- CFs in BOMs



- CFs in WOMs
  - *Inter-word* CFs & *intra-word* CFs



# Converting March Tests

- Any given BOM march test can be converted to a WOM march test
  - With additional tests to cover intra-word faults
- A WOM march test is a concatenation of two march tests
  - {Inter-word march test, intra-word march test}
- The inter-word march test can directly be obtained from the BOM march test
  - Replace the bit-operation "r0", "w0", "r1", and "w1" with the word-operation "rD", "wD", "rD'", and "wD'", where D is called *data background*

# Converting March Tests

- The intra-word faults can be detected by *a single march element* with *different operations* and *data backgrounds*
  - E.g., intra CFst can be covered by $(wd_1, rd_1, ..., wd_n, rd_n)$ with various data backgrounds (DBs)
  - Note that the DBs can be applied in any order
- The above intra-word test can be modified as follows, without any impact on the fault coverage
  - Extra Read operations can be added
  - The single march element can be divided into any number of march elements, and for each march element the addressing order can be chosen freely

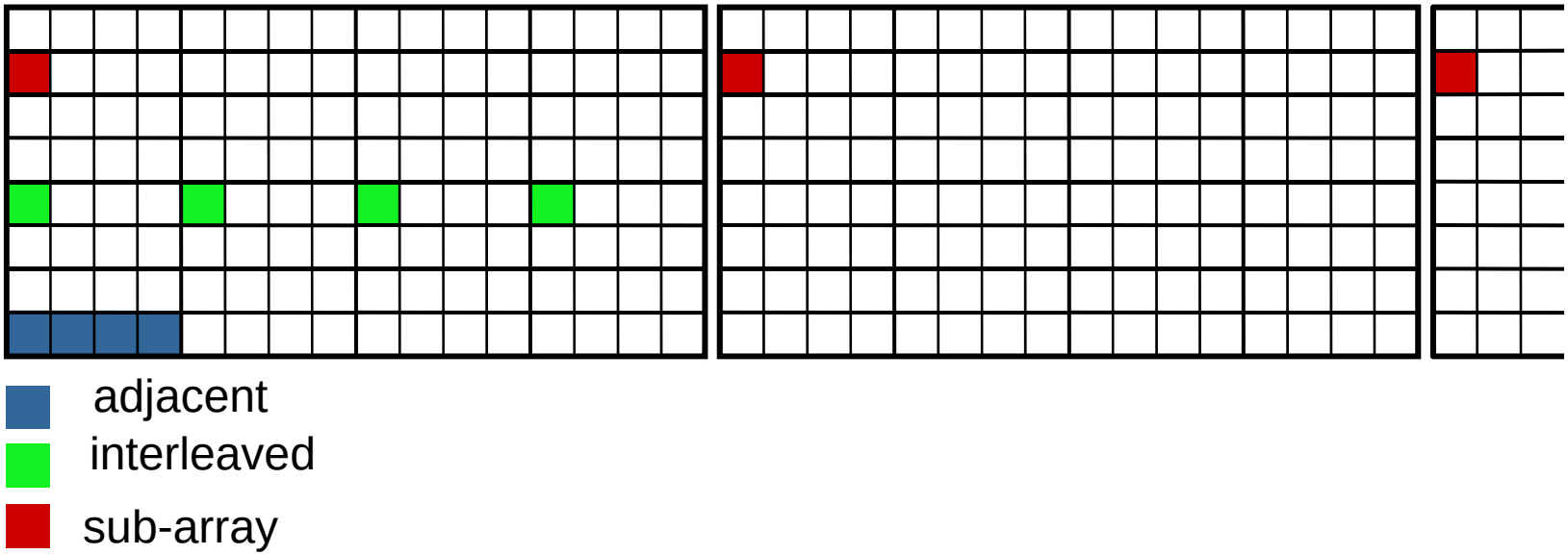
# Compact WOM Tests

- If you have a bit-oriented march test, then you can obtain a compact WOM test with
  - Replace the bit-operation "r0", "w0", "r1", and "w1" with all-0 and all-1 data backgrounds
  - Concatenate a march element $(wd, w\overline{d}, r\overline{d}, wd, rd)$ for d={0101..01,0011..11, …}
- For example, the March C- can be extended as follows to test a memory with 4-bit words

$$\{\Updownarrow(w0000);\Uparrow(r0000,w11111);\Uparrow(r11111,w0000);$$
$$\Downarrow(r0000,w1111);\Downarrow(r1111,w0000);\Updownarrow(r0000);$$
$$\Updownarrow(w0101,w1010,r1010,w0101,r0101);$$
$$\Updownarrow(w0011,w1100,r1100,w0011,r0011)\}$$

# Memory Organization & WOM Tests

- WOMs can be organized internally in many different ways
  - Adjacent; interleaved; sub-arrays

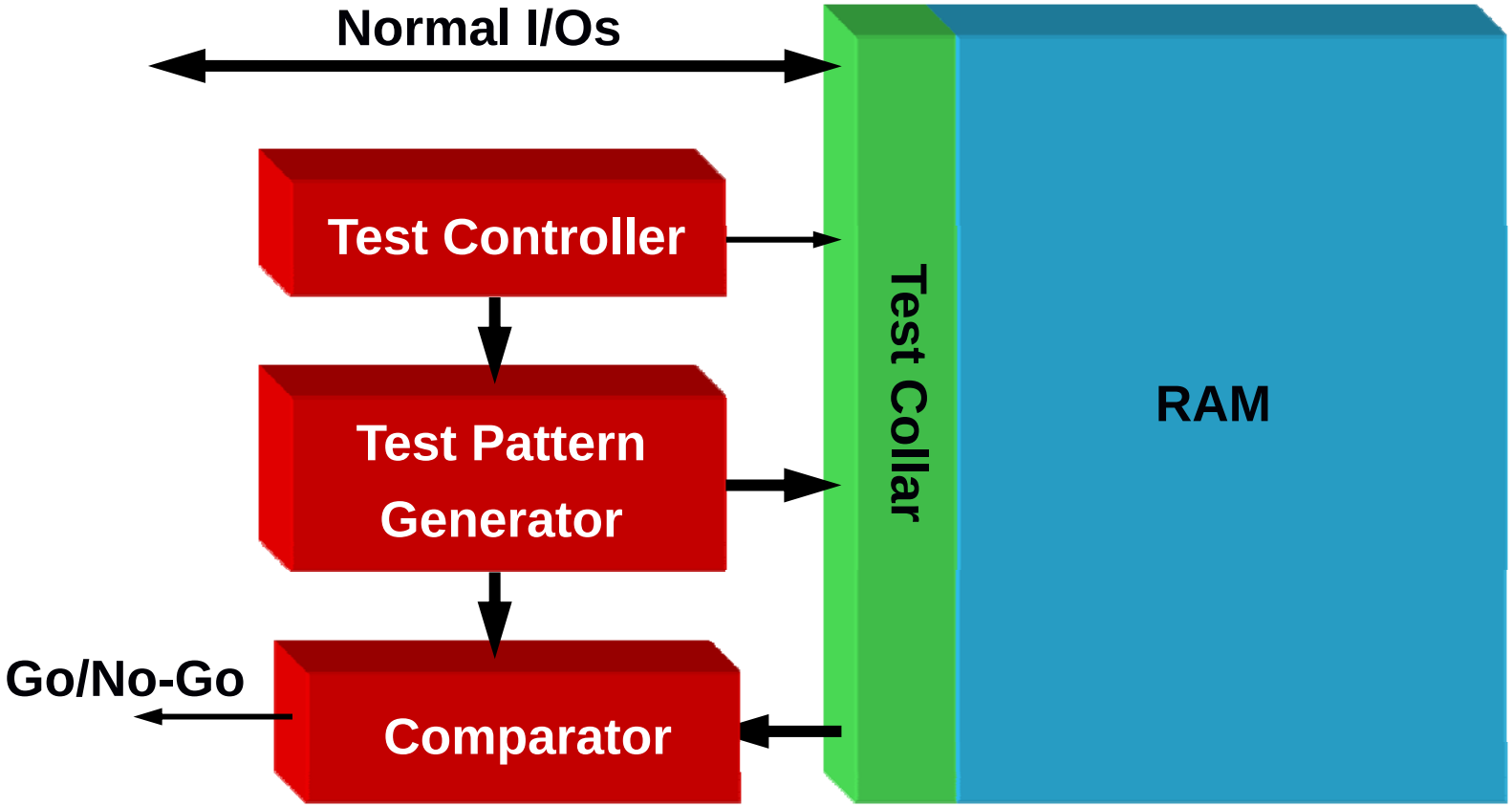



  - For sub-array organized WOMs, the BOM tests for CFs will detect the CFs within a B-bit word such that no intra-word tests are required

# RAM BIST

- A typical RAM BIST consists of the following components
  - Controller, test pattern generator, and comparator
- Controller
  - Generate control signals to the test pattern generator & the memory under test
- Test pattern generator (TPG)
  - Generate the required test patterns and Read/Write signals
- Comparator
  - Evaluate the response

# Typical RAM BIST Architecture

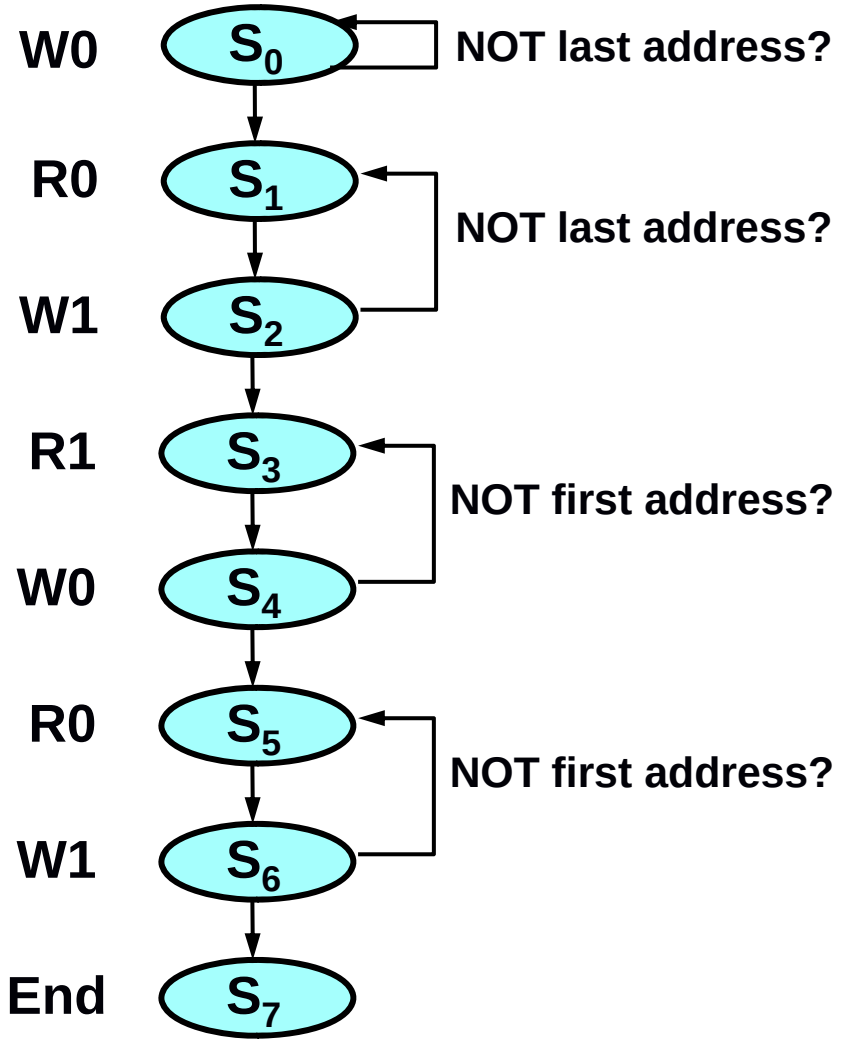


# RAM BIST

- In general, two BIST approaches have been proposed for the RAMs
  - FSM-based RAM BIST
  - ROM-based RAM BIST
- Controller
  - Generate control signals to the test pattern generator & the memory under test
- Test pattern generator (TPG)
  - Generate the required test patterns and Read/Write signals
- Comparator
  - Evaluate the response

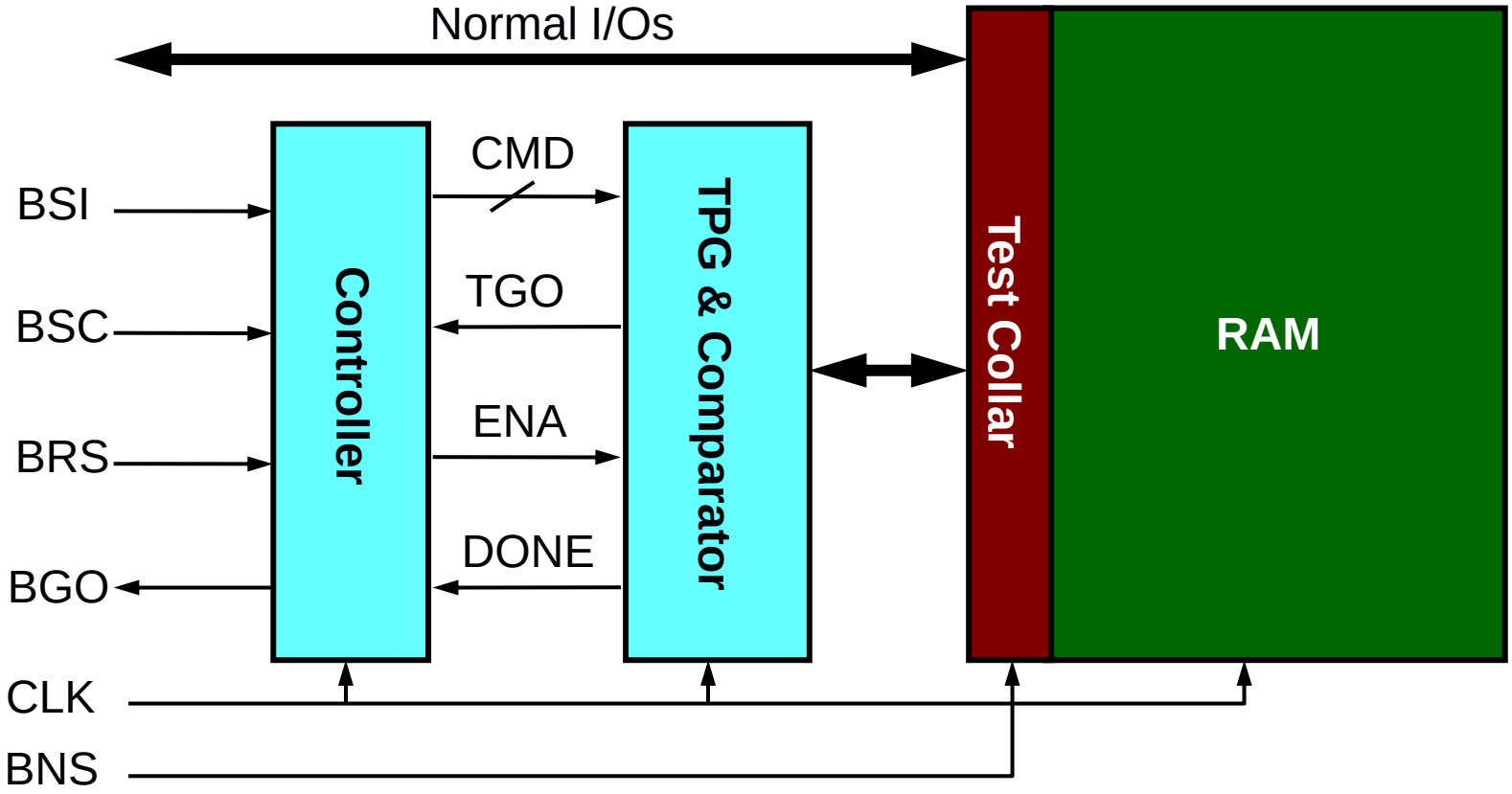
# Controller

- An example of the state diagram of controller

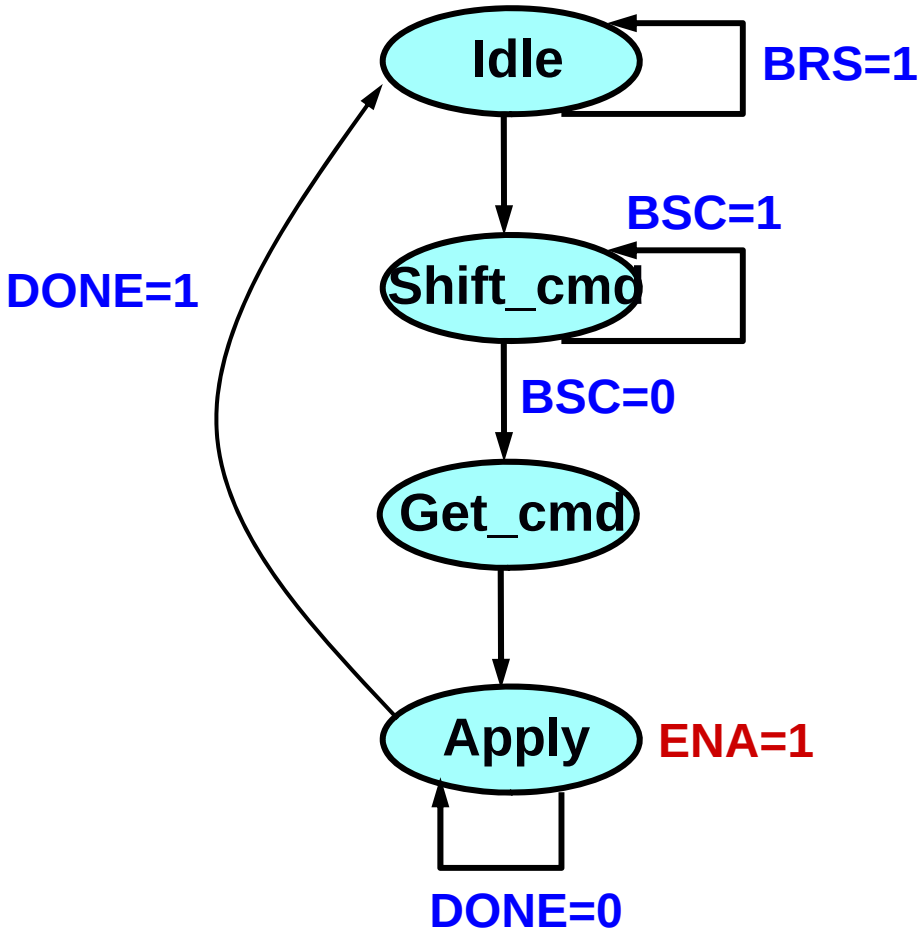


# Programmable RAM BIST

- An example of the programmable RAM BIST



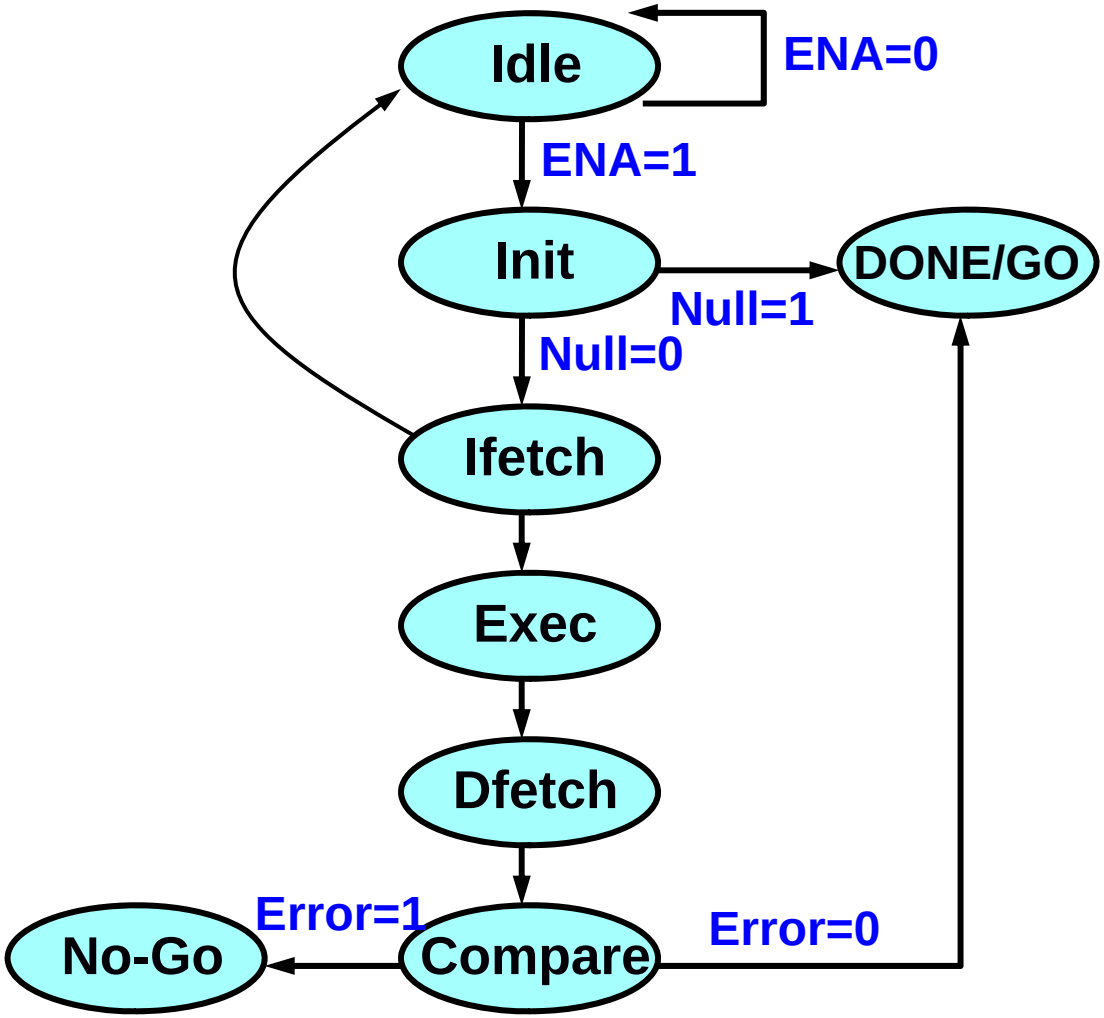
# FSM State Diagram of the Controller



# Programmability

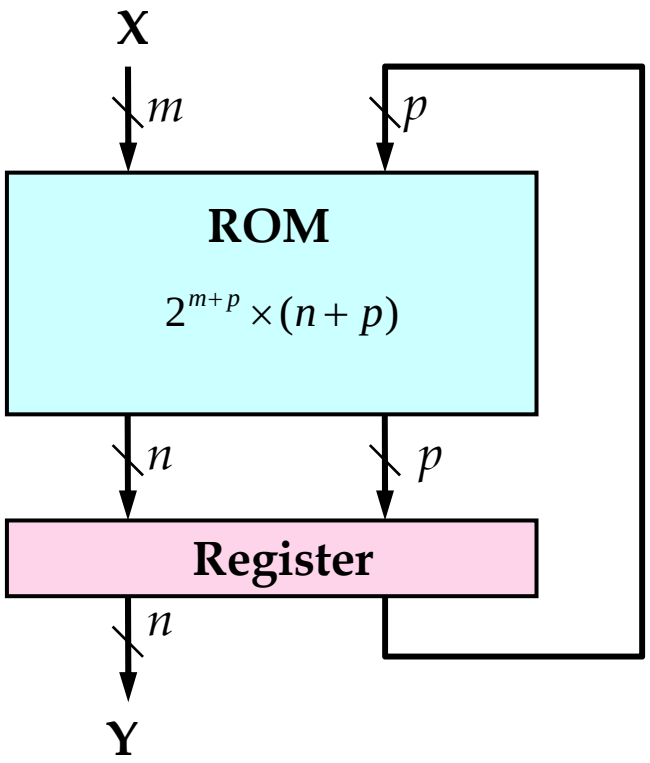
- The programmability can be achieved by using test command
- The test command format
  - | U/D | OP | Data background |
    |---|---|---|
  - U/D: ascending/descending address sequence
  - OP: test operations
    - For example, wa, rawa', rawa'ra, warawa'ra', etc.
  - Data backgrounds
- The width of each field affects the programmability of the BIST design
  - For example, if 4 bits are used for OP, then only 16 possible test operations can be generated

# FSM State Diagram of the TPG



# ROM-Based BIST

- A typical ROM model




Source: R. Senhadji-Navarro, et al., Electronic Letters, 2004

# ROM-Based BIST

- A typical ROM model for March test elements

| Address | CEN | WEN | Invert | Addr_Incr | State | March element |
|---|---|---|---|---|---|---|
| 000_000 | 0 | 0 | 0 | 0 | 000 | Idle |
| 001_000 | 1 | 0 | 0 | 1 | 000 | wa |
| 010_000 | 1 | 1 | 0 | 1 | 000 | ra |
| 011_000 | 1 | 1 | 0 | 0 | 001 | ra |
| 011_001 | 1 | 0 | 1 | 1 | 000 | wa′ |
| 100_000 | 1 | 1 | 1 | 0 | 001 | ra′ |
| 100_001 | 1 | 0 | 0 | 1 | 000 | wa |

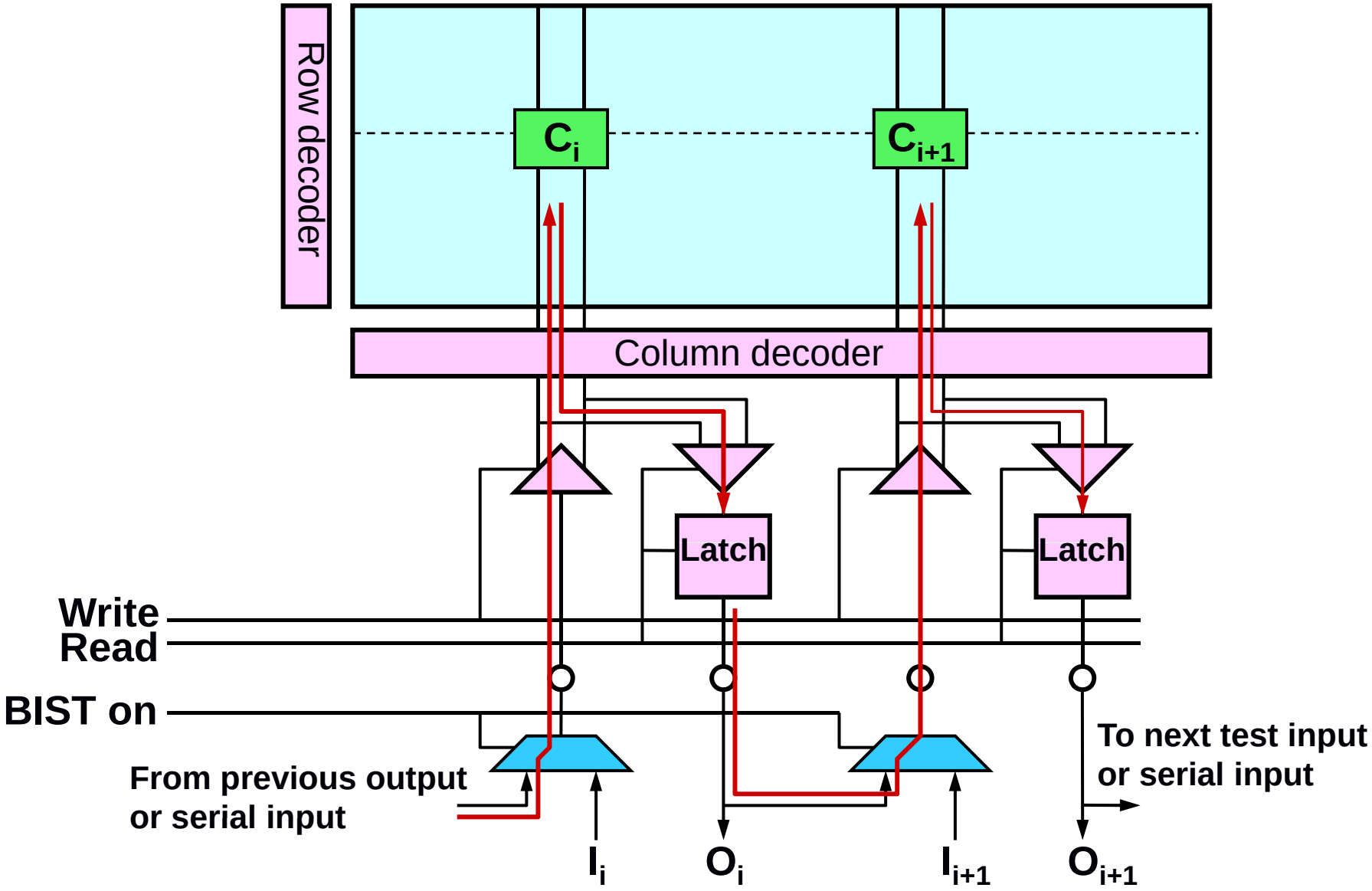
# Serial BIST

- Today's telecommunication ICs often have a variety of multi-port memories on one chip
- Typical RAM BISTs evaluate all the bits of a memory word in parallel as it is read
- We can encounter significant problems when applying these BIST schemes to chips that have multiple embedded RAMs of *varying sizes* and *port configurations*
  - The area cost of these BIST designs would be unacceptably high
- One better solution is a serial BIST technique
  - To share BIST design among several RAMs

# Benefits of Serial BIST

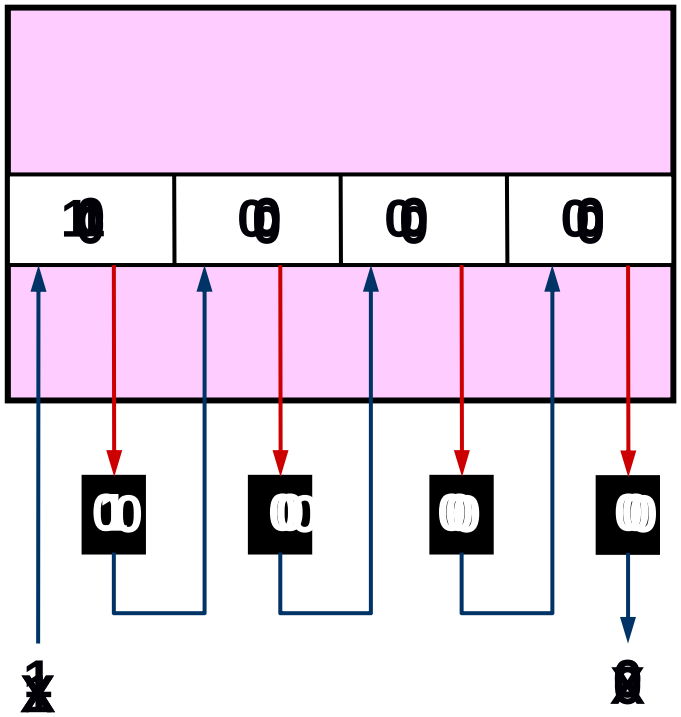
- Only a small amount of additional circuitry is required
- Only a few lines are needed to connect the RAM to the test controller
- Several RAM blocks easily share the BIST controller hardware
- The serial-access mode does not compromise the RAM cycle time
- Existing memory designs do not need any modification to use the serial interface

# Serial-Data-Path Connection



# Serial Shift Operation

- An example of serial shift operations for the match element (*R0W1*)

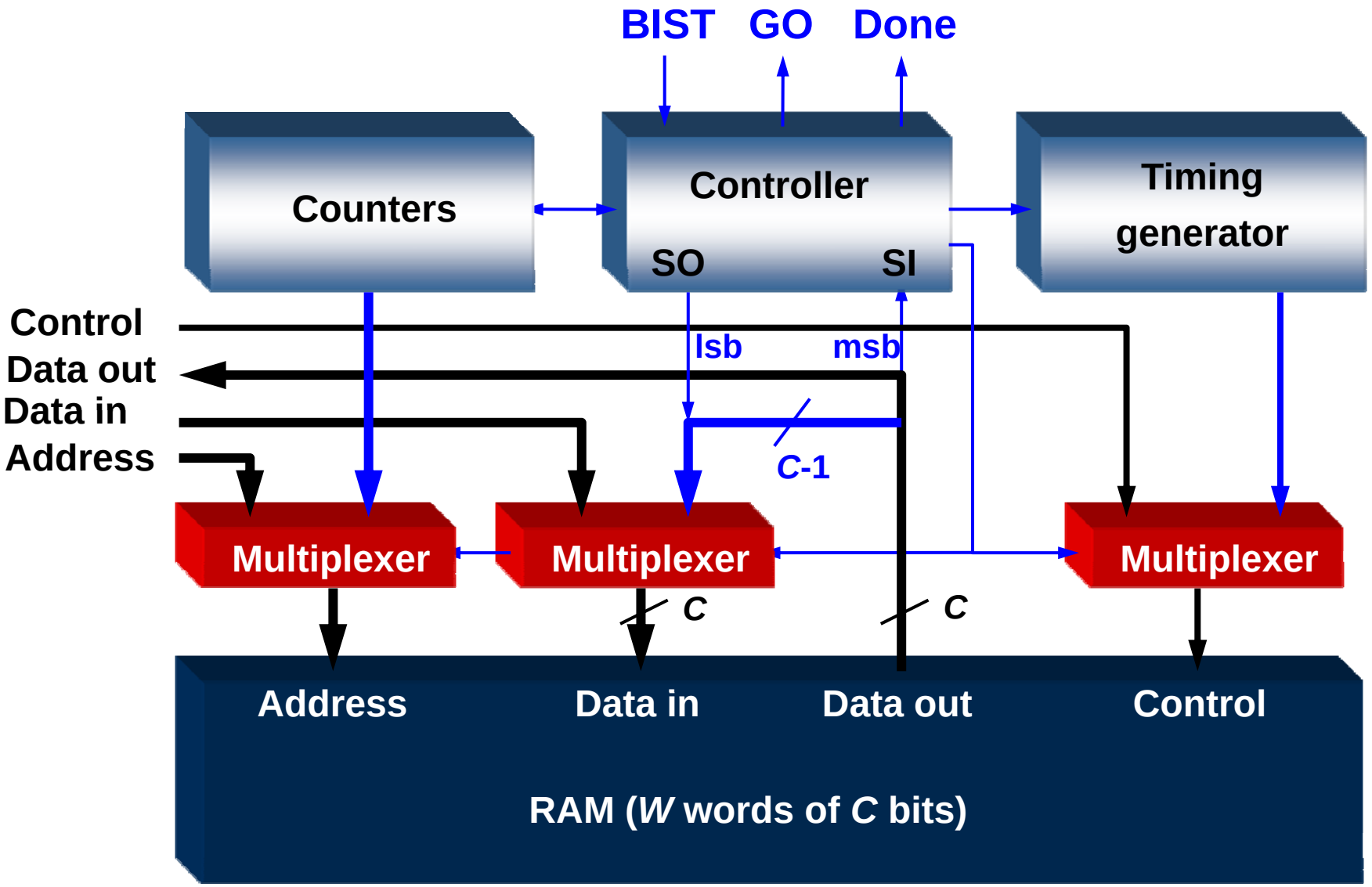


| Time | Operation | Serial in | Word content | | | | Serial out |
|---|---|---|---|---|---|---|---|
| 0 | | X | 0 | 0 | 0 | 0 | X |
| 1 | R0 | X | 0 | 0 | 0 | 0 | 0 |
| 2 | W1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 3 | R0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 4 | W1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 5 | R0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 6 | W1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 7 | R0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 8 | W1 | 1 | 1 | 1 | 1 | 1 | 0 |

# Serial March (SMarch)

- Assume that a RAM has *W* words, and each word contains *C* bits
- A Read operation is denoted by *R0*, *R1*, or *Rx*, depending on the expected value at the serial output (*x*=don't care)
- For a write operation, the terms *W0* or *W1* are used and only the serial input is forced to the value indicated
- The SMarch modified from March C- is as follows
  - $\Updownarrow (RxW\ 0)^C (R0,W0)^C; \Uparrow (R0,W1)^C (R1,W1)^C$
    $\Uparrow (R1,W0)^C (R0,W0)^C; \Downarrow (R0,W1)^C (R1,W1)^C$
    $\Downarrow (R1,W0)^C (R0,W0)^C; \Downarrow (R0,W0)^C (R0,W0)^C$

# Serial BIST Architecture



# References


- [1] M. Sachdev,"Defect Oriented Testing for CMOS Analog and Digital Circuits", Kluwer Academic, 1999.
- [2] M.-L. Bushnell and V.-D. Agrawal,"Essentials of Electronic Testing for Digital, Memory & Mixed-Signal VLSI Circuits", Kluwer Academic, 2000.
- [3] A. J. van de Goor and C. A. Verruijt,"An Overview of Deterministic Functional RAM Chip Testing", ACM Computing Surveys, vol. 22, no. 1, March 1990.
- [4] A. J. van de Goor, I. B. S. Tlili, and S. Hamdioui,"Converting March Tests for Bit-Oriented Memories into Tests for Bit-Oriented Memories", MTDT, pp. 46-52, 1998.
- [5] A. J. van de Goor and G. N. Gaydadjiev,"March U: a test for unlinked memory faults", IEE Proc. Circuit Devices Syse., vol.144, no. 3, pp.155-160, 1997.
- [6] D. S. Suk and S. M. Reddy," A march test for functional faults in semiconductor random-access memories" IEEE Trans. Comput., C-30, 12, pp.982-985, 1981.
- [7] K. L. Cheng, M. F. Tsai, and C. W. Wu,"Neighborhood pattern-sensitive fault testing and diagnostics for random-access memories", IEEE Trans. CAD, vol.21, no.11, pp. 1328-1336, nov., 2002.
- [8]C. F. Wu, C. T. Huang, and C. W. Wu"RAMSES: a fast memory fault simulator," DFT99, pp.199-202, 1996.
- [9] B. Nadeau-Dostie, A. Silburt, and V. K. Agarwal,"Serial interfacing for embedded-memory testing", IEEE D&T, pp.52-63, apr. 1990.

# References


- [10] C. W. Wu,"VLSI testing & design for testability II: Memory built-in self-test", http://larc.ee.nthu.edu.tw/~cww/
- [11] A. J. van de Goor and C. A. Verruijt,"An Overview of Deterministic Functional RAM Chip Testing", ACM Computing Surveys, vol. 22, no. 1, March 1990.
- [12] A. K. Sharma,"Semiconductor Memories – Technology, Testing, and Reliability", IEEE PRESS, 1997.
- [13] R. Dekker, F. Beenker, and L. Thijssen,"Fault Modeling and Test Algorithm Development for Static Random Access Memories", Int. Test Conf., pp. 343-352, 1988.
- [14] M. Sachdev,"Defect Oriented Testing for CMOS Analog and Digital Circuits", Kluwer Academic, 1998.
- [15] B. F. Cockburn,"Tutorial on Semiconductor Memory Testing", JETTA, pp. 321-336, 1994.