Chapter 9

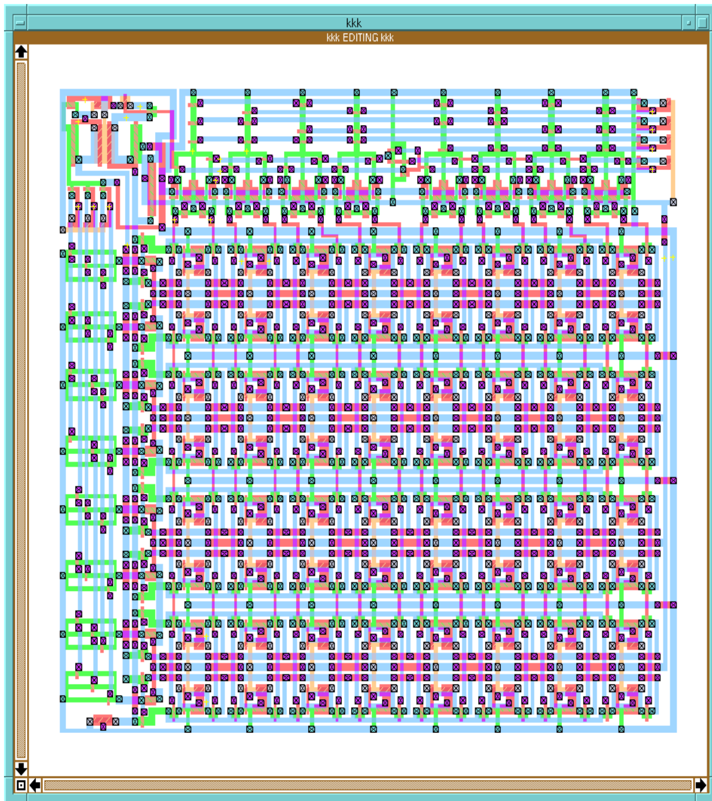
Memory Diagnosis and Built-In Self-Repair

What is this chapter about?

- Why diagnostics?
  - Yield improvement
    - Repair and/or design/process debugging
- BIST design with diagnosis support
- MECA: a system for automatic identification of fault site and fault type
- Built-in self-repair (BISR) for embedded memories
  - Redundancy analysis (RA) algorithms
  - Built-in redundancy analysis (BIRA)

How to Identify Faults?

RAM Circuit/Layout



Tester/BIST Output

```
Test  1:  4:Stage 4, :Functional  (294913 Cyc) Cat:

 S:Pattern            Start-Loc  Stop-Loc    Size Result
-------------------------------------------------------
 1:</fs80a009.avf          37    294949    294913 *FAIL*
 S          PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP
 i          ____________________________________
 t          000000000000000000000000000000000000
 e          000000000111111111122222222233333333
 #          123456789012345678901234678901246789
 1:   85507 11001010011010010110.0L0HL0.0.0.0L0L
 1:   85511 11001010011110010110.0L0HL0.0.0.0L0L
 1:   85515 11001010111010010110.0L0HL0.0.0.0L0L
 1:   85519 11001010111110010110.0L0HL0.0.0.0L0L
        .
        .
 1: 242163 11001010111110000111H1/1L/1H1H1H1/1/
 1: 242167 11001010111010000111H1/1L/1H1H1H1/1/
 1: 242171 11001010011110000111H1/1L/1H1H1H1/1/
 1: 242175 11001010011010000111H1/1L/1H1H1H1/1/
        .
        .
```

Fault Model Subtypes

Name	Agr	Vtm	Addr
SAF_0	-	1/0	-
SAF_1	-	0/1	-
TF_0	-	$\downarrow$/1	-
TF_1	-	$\uparrow$/0	-
$CFin_0$	$\downarrow$	$\updownarrow$	$A < V$
$CFin_1$	$\downarrow$	$\updownarrow$	$A > V$
$CFin_2$	$\uparrow$	$\updownarrow$	$A < V$
$CFin_3$	$\uparrow$	$\updownarrow$	$A > V$
$CFst_0$	0	1/0	$A < V$
$CFst_1$	0	1/0	$A > V$
$CFst_2$	0	0/1	$A < V$
$CFst_3$	0	0/1	$A > V$
$CFst_4$	1	1/0	$A < V$
$CFst_5$	1	1/0	$A > V$

Name	Agr	Vtm	Addr
$CFst_6$	1	0/1	$A < V$
$CFst_7$	1	0/1	$A > V$
$CFid_0$	$\downarrow$	1/0	$A < V$
$CFid_1$	$\downarrow$	1/0	$A > V$
$CFid_2$	$\downarrow$	0/1	$A < V$
$CFid_3$	$\downarrow$	0/1	$A > V$
$CFid_4$	$\uparrow$	1/0	$A < V$
$CFid_5$	$\uparrow$	1/0	$A > V$
$CFid_6$	$\uparrow$	0/1	$A < V$
$CFid_7$	$\uparrow$	0/1	$A > V$
AF_0	-	-	$A < V$
AF_1	-	-	$A > V$
SOF	-	-	-

March Signature & Dictionary

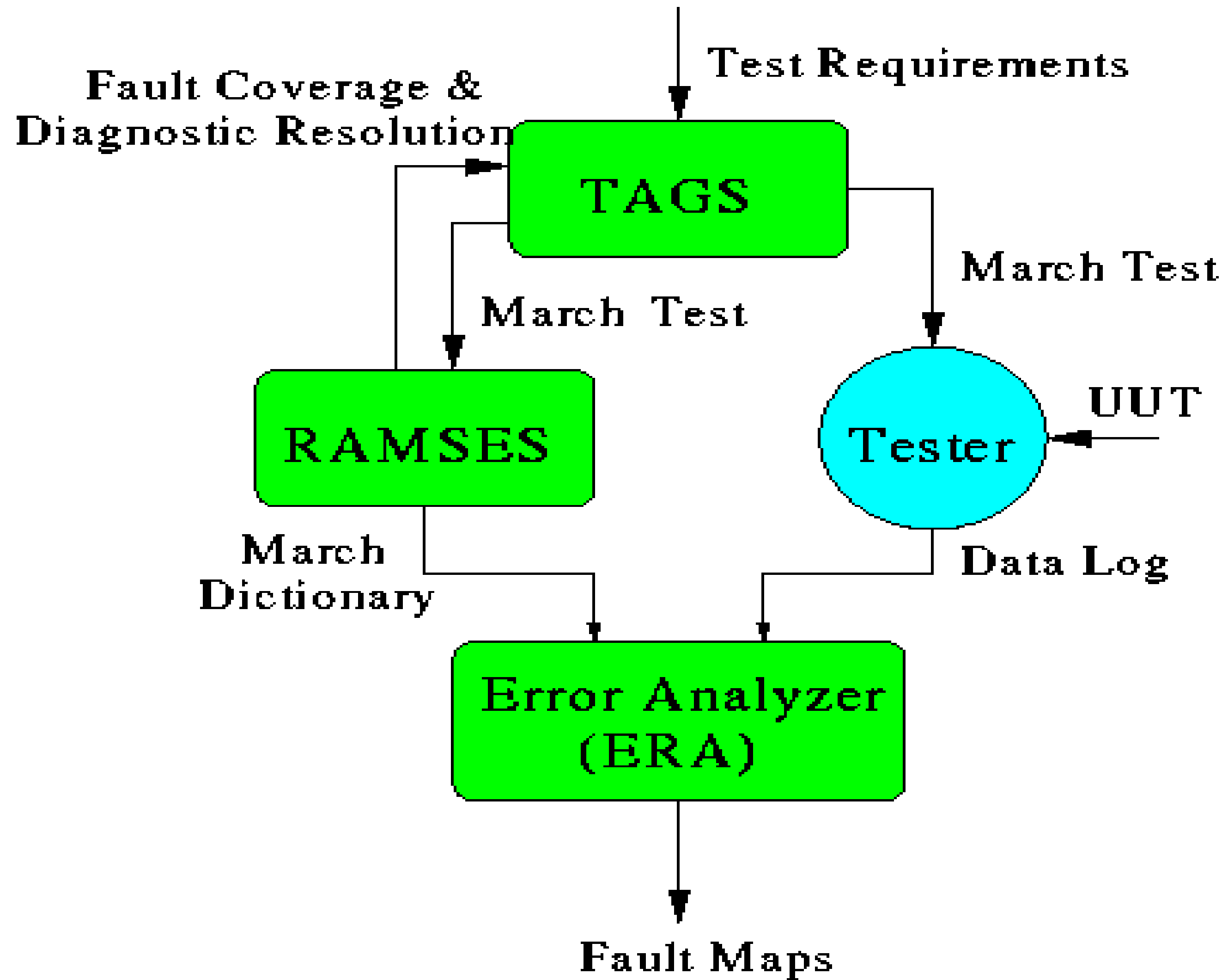
March 11N

$$\Updownarrow(w0)\Uparrow(r0,w1)\Updownarrow(r1)\Uparrow(r1,w0)\Downarrow(r0,w1)\Downarrow(r1,w0)\Updownarrow(r0)$$

E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

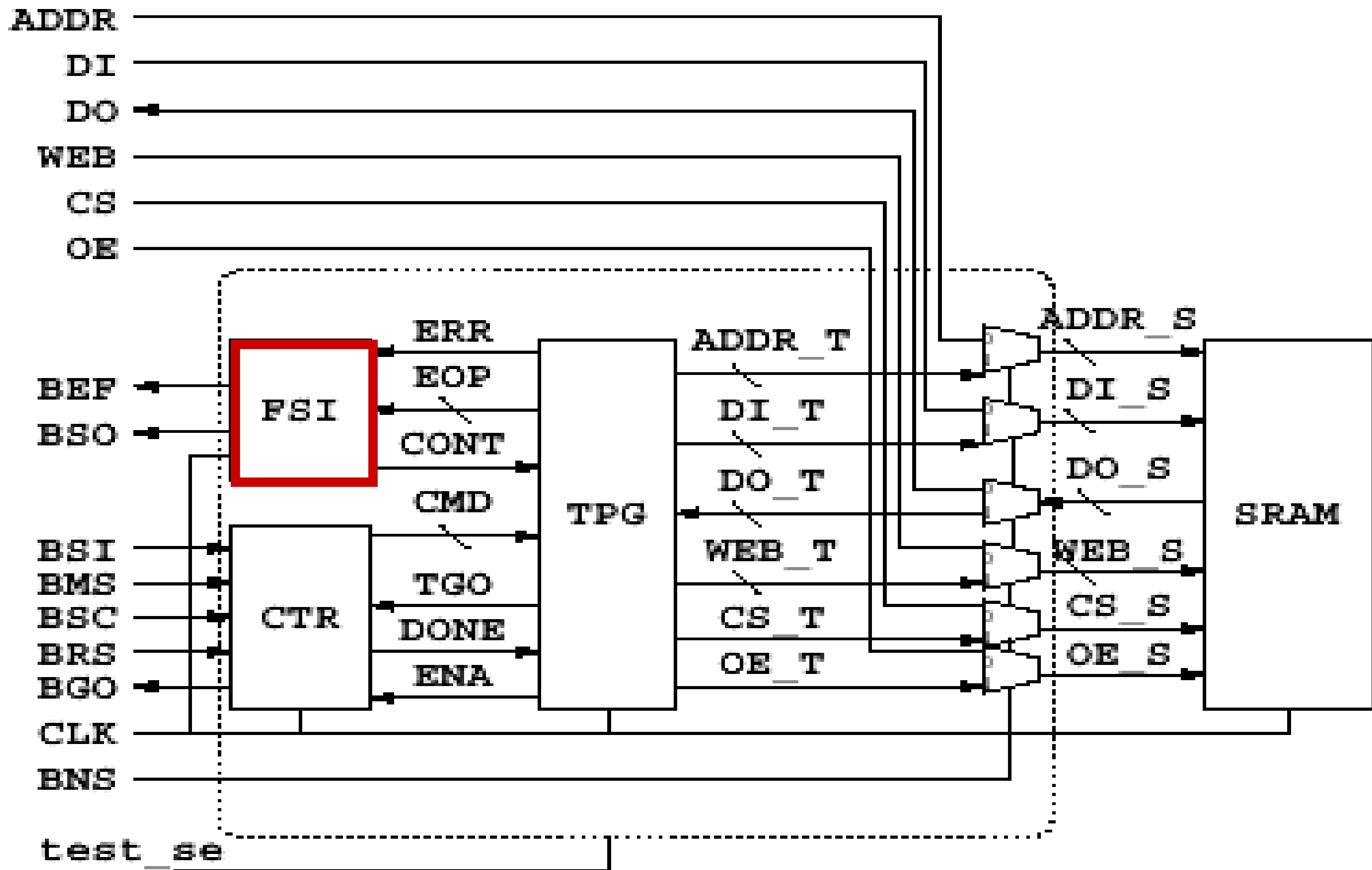
Fault/Error Bitmap	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}
SAF_0	0	0	0	1	1	0	0	0	1	0	0
SAF_1	0	1	0	0	0	0	1	0	0	0	1
$CFin_0$	0	0	0	0	1	0	0	0	0	0	1
$CFin_1$	0	0	0	0	0	0	1	0	1	0	0
$CFin_2$	0	1	0	0	0	0	0	0	1	0	0
$CFin_3$	0	0	0	1	1	0	1	0	0	0	0
$CFst_0$	0	0	0	0	1	0	0	0	0	0	0
$CFst_1$	0	0	0	0	0	0	0	0	1	0	0
$CFst_2$	0	0	0	0	0	0	1	0	0	0	1
$CFst_3$	0	1	0	0	0	0	0	0	0	0	1
$CFst_4$	0	0	0	1	0	0	0	0	1	0	0
$CFst_5$	0	0	0	1	1	0	0	0	0	0	0
$CFst_6$	0	1	0	0	0	0	0	0	0	0	0
$CFst_7$	0	0	0	0	0	0	1	0	0	0	0

Memory Error Catch and Analysis (MECA)



Source: Wu, *et al.*, ICCAD00

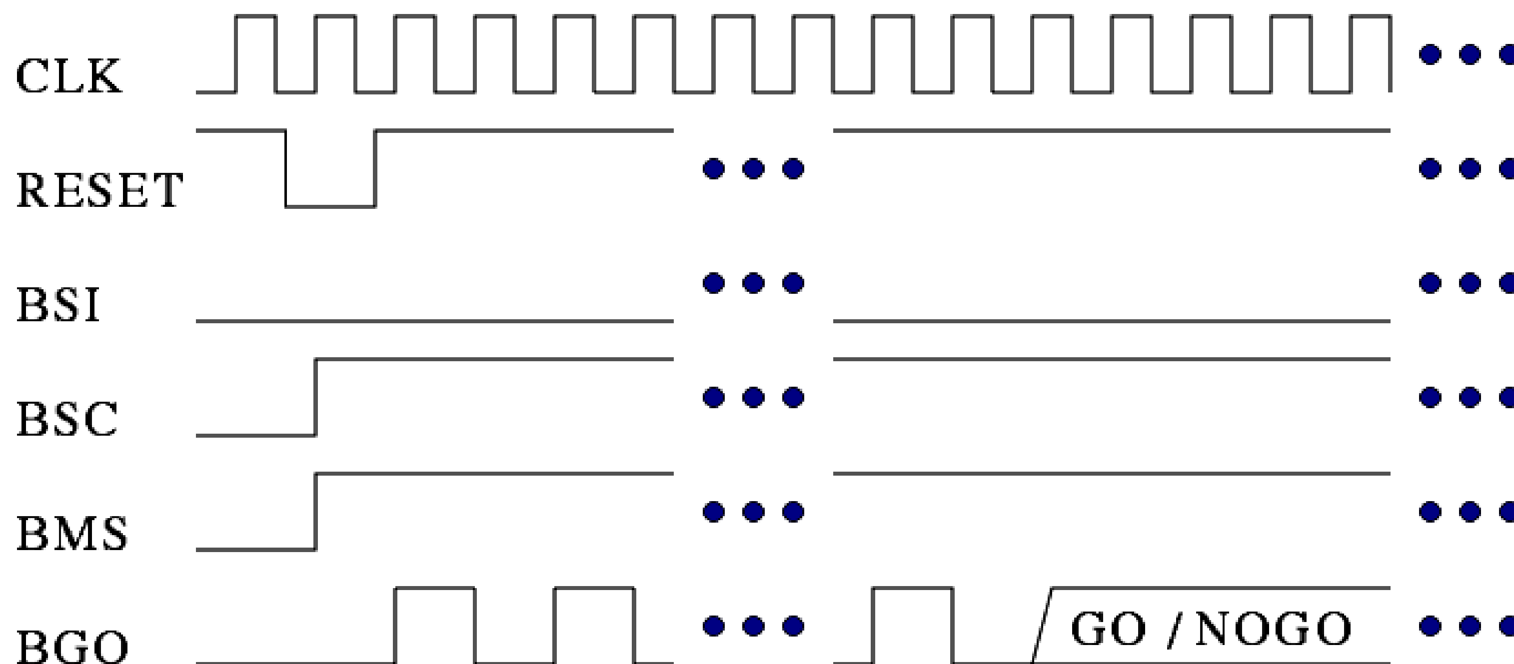
BIST with Diagnosis Support



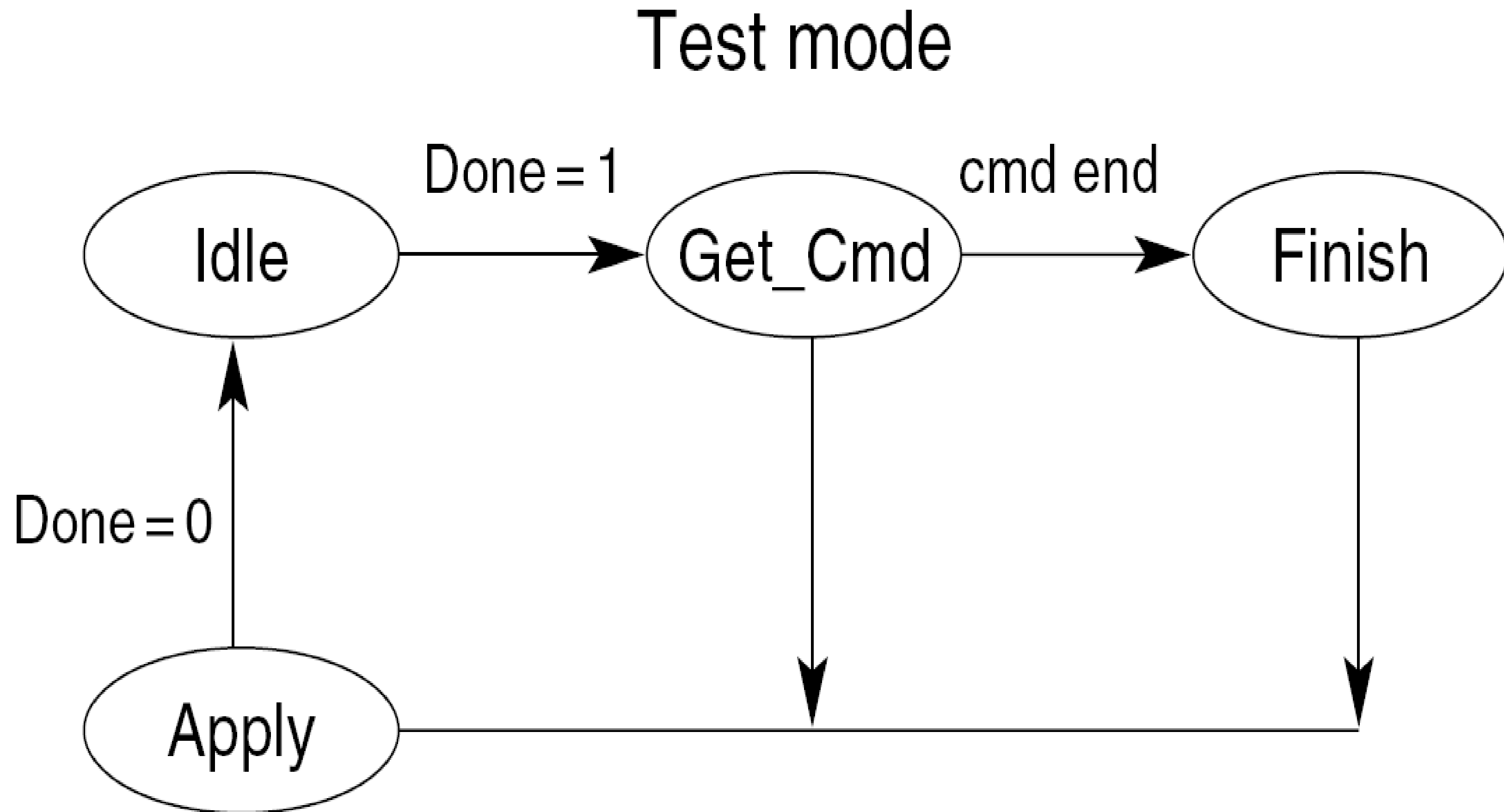
Source: Wang, *et al.*, ATS00

Test Mode

- In Test Mode it runs a fixed algorithm for production test and repair.
  - Only a few pins need to be controlled, and *BGO* reports the result (Go/No-Go).



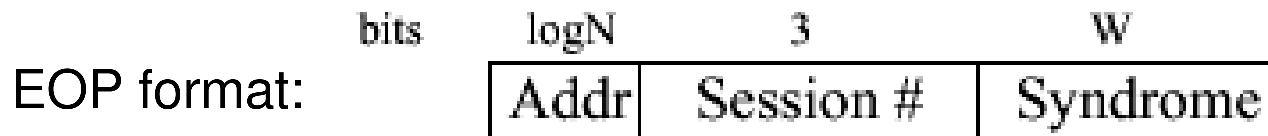
CTR State Diagram in Test Mode

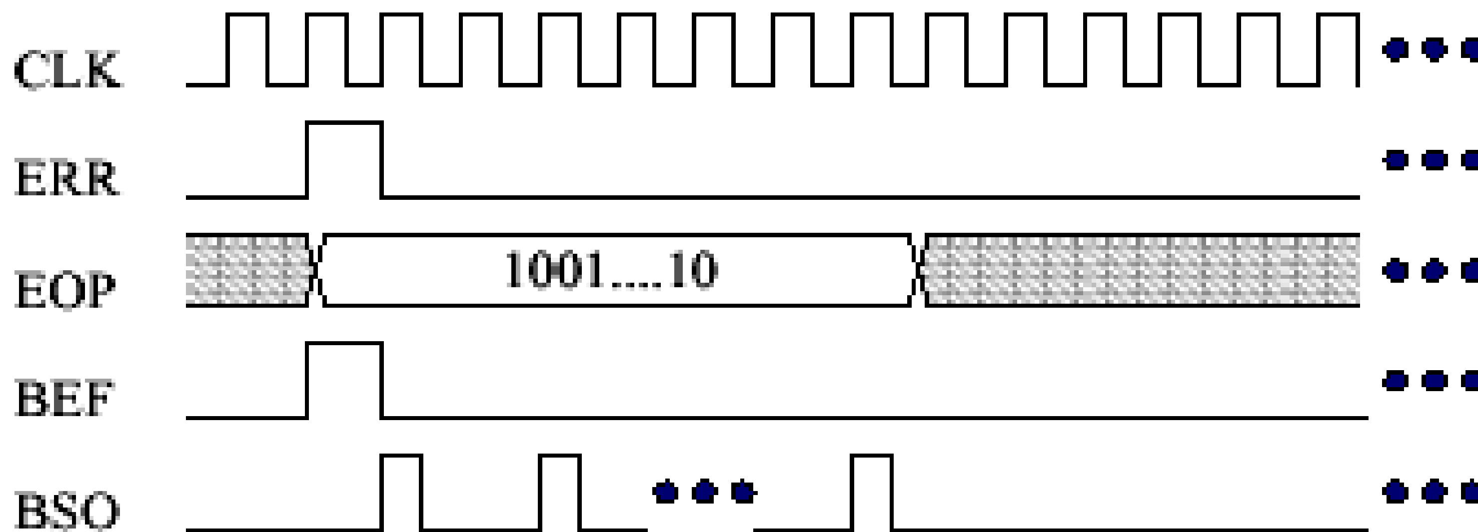


Fault Analysis Mode (FSI Timing)

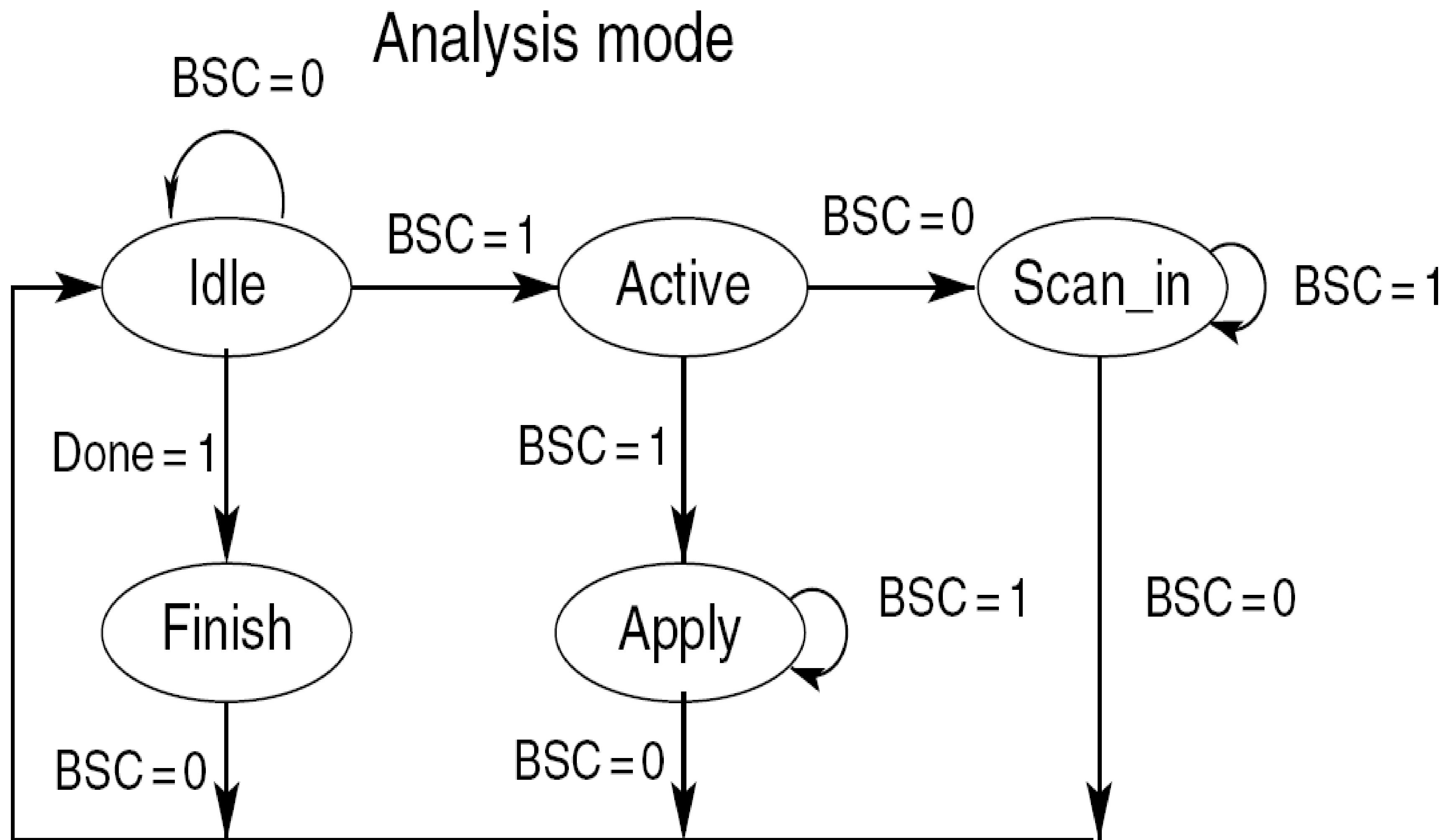
- In Fault Analysis Mode, we can apply a longer March algorithm for diagnosis
  - FSI captures the error information of the faulty cells

EOP format:





CTR State Diagram in Analysis Mode



Fault Analysis

- Derive analysis equations from the fault dictionary

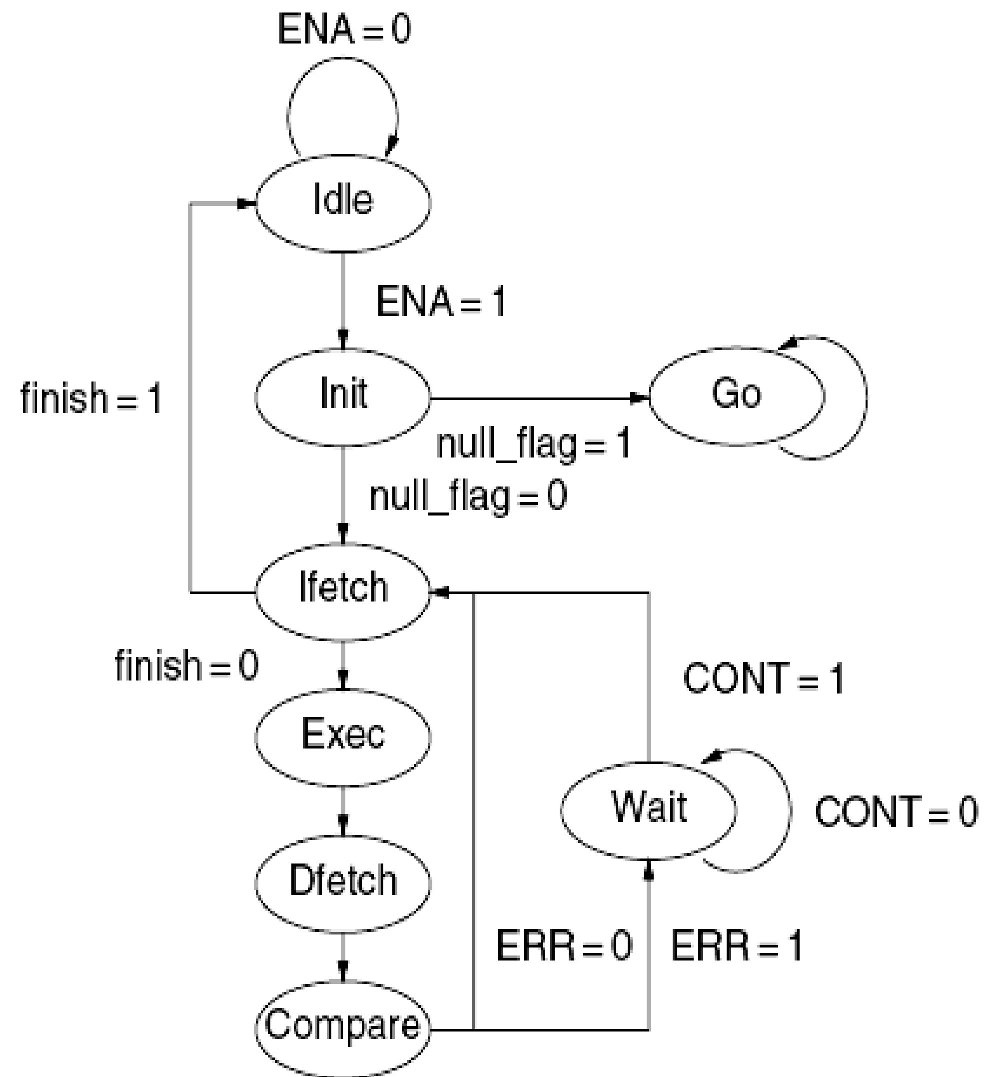
Fault/Error Bitmap	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}
SAF_0	0	0	0	1	1	0	0	0	1	0	0
SAF_1	0	1	0	0	0	0	1	0	0	0	1

- Convert error maps to fault maps by the equations

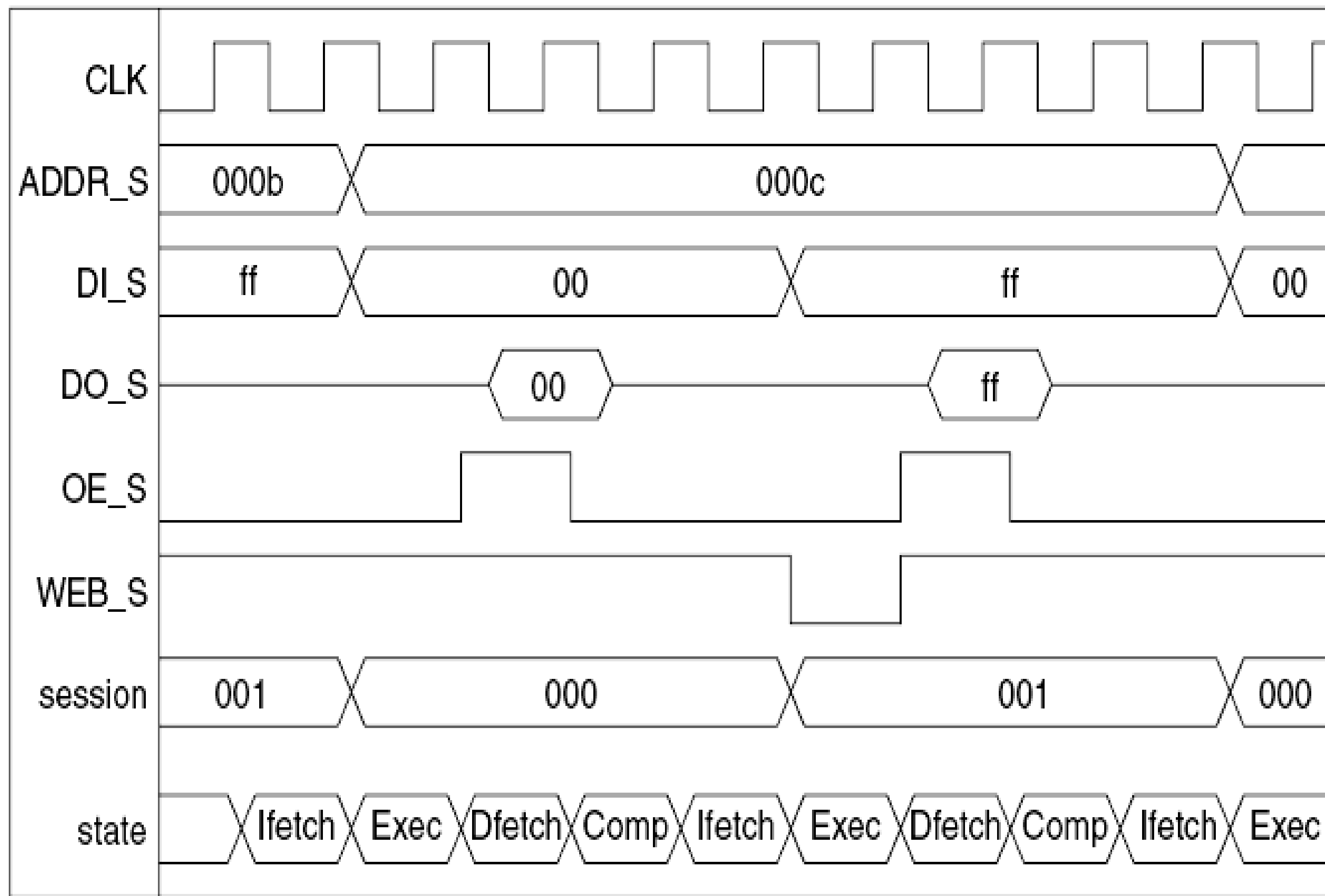
$$F_{SAF_0} = \overline{E_0} \cap \overline{E_1} \cap \overline{E_2} \cap E_3 \cap E_4 \cap \overline{E_5} \cap \overline{E_6} \cap \overline{E_7} \cap E_8 \cap \overline{E_9} \cap \overline{E_{10}}.$$

$$F_{SAF_1} = \overline{E_0} \cap E_1 \cap \overline{E_2} \cap \overline{E_3} \cap \overline{E_4} \cap \overline{E_5} \cap E_6 \cap \overline{E_7} \cap \overline{E_8} \cap \overline{E_9} \cap E_{10}.$$

TPG State Diagram



Waveform Generated by TPG



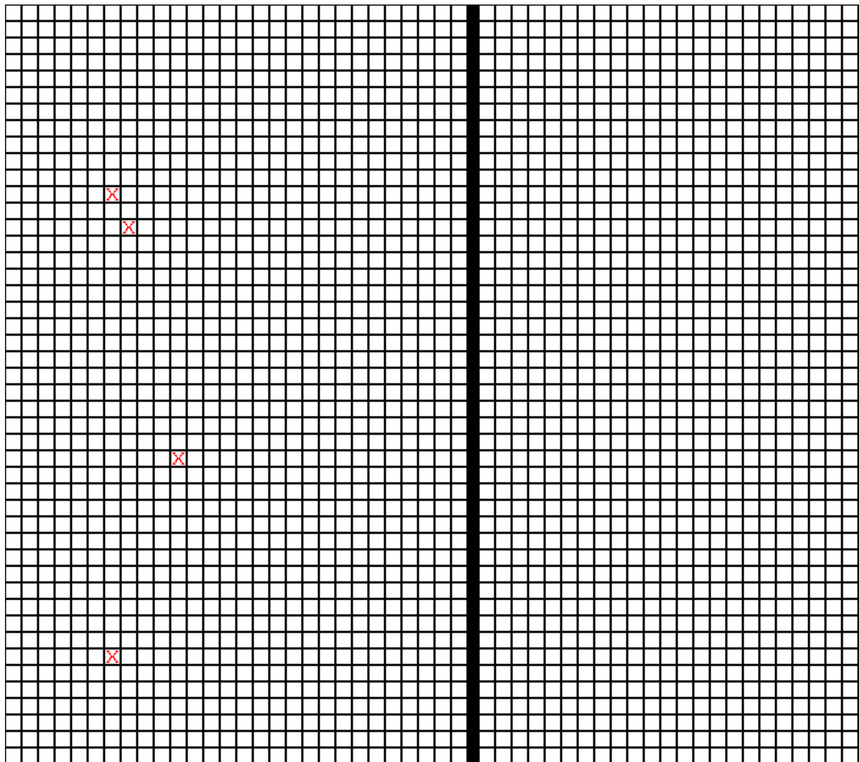
Diagnostic Test Algorithm Generation

- Start from a base test: generated by TAGS, or user-specified
- Generation options reduced to Read insertions
- Diagnostic resolution: percentage of faults that can be distinguished

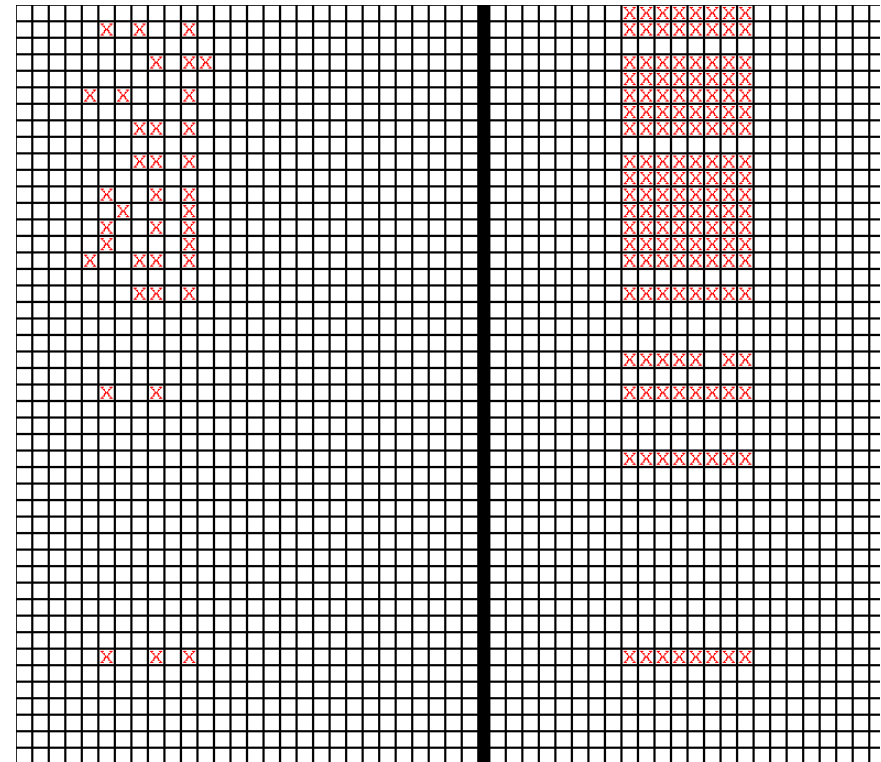
$T(N)$	Resolution	March algorithm
$11N$	0.586	⇑(w0) ⇑(r0_w1) ⇑(r1_w0) ⇓(r0_w1) ⇓(r1_w0_r0) ⇑(r0)
$12N$	0.690	⇑(w0) ⇑(r0_w1) ⇑(r1_w0) ⇓(r0_w1) ⇑(r1) ⇓(r1_w0_r0) ⇑(r0)
$13N$	0.793	⇑(w0) ⇑(r0_w1_r1) ⇑(r1_w0) ⇓(r0_w1) ⇑(r1) ⇓(r1_w0_r0) ⇑(r0)
$14N$	0.828	⇑(w0) ⇑(r0_w1) ⇑(r1_w0_r0) ⇑(r0) ⇓(r0_w1) ⇑(r1) ⇓(r1_w0_r0) ⇑(r0)
$14N$	0.828	⇑(w0) ⇑(r0_w1) ⇑(r1) ⇑(r1_w0_r0) ⇓(r0_w1_r1) ⇓(r1_w0_r0) ⇑(r0)
$14N$	0.828	⇑(w0) ⇑(r0_w1) ⇑(r1) ⇑(r1_w0) ⇑(r0) ⇓(r0_w1_r1) ⇓(r1_w0_r0) ⇑(r0)
$14N$	0.828	⇑(w0) ⇑(r0_w1_r1) ⇑(r1_w0) ⇓(r0_w1_r1) ⇑(r1) ⇓(r1_w0_r0) ⇑(r0)
$15N$	0.897	⇑(w0) ⇑(r0_w1_r1) ⇑(r1_w0_r0) ⇑(r0) ⇓(r0_w1) ⇑(r1) ⇓(r1_w0_r0)⇑(r0)
$15N$	0.897	⇑(w0) ⇑(r0_w1) ⇑(r1) ⇑(r1_w0_r0) ⇑(r0) ⇓(r0_w1_r1) ⇓(r1_w0_r0) ⇑(r0)
$15N$	0.897	⇑(w0) ⇑(r0_w1) ⇑(r1) ⇑(r1_w0_r0) ⇑(r0) ⇓(r0_w1) ⇑(r1) ⇓(r1_w0_r0) ⇑(r0)
$16N$	0.931	⇑(w0) ⇑(r0_w1_r1) ⇑(r1) ⇑(r1_w0_r0) ⇑(r0) ⇓(r0_w1) ⇑(r1) ⇓(r1_w0_r0) ⇑(r0)
$16N$	0.931	⇑(w0) ⇑(r0_w1_r1) ⇑(r1) ⇑(r1_w0_r0) ⇑(r0) ⇓(r0_w1_r1) ⇓(r1_w0_r0) ⇑(r0)
$16N$	0.931	⇑(w0) ⇑(r0_w1_r1) ⇑(r1) ⇑(r1_w0) ⇑(r0) ⇓(r0_w1_r1) ⇑(r1) ⇓(r1_w0_r0) ⇑(r0)
$17N$	0.966	⇑(w0) ⇑(r0_w1_r1) ⇑(r1) ⇑(r1_w0_r0) ⇑(r0) ⇓(r0_w1_r1) ⇑(r1) ⇓(r1_w0_r0) ⇑(r0)

Fault Bitmap Examples

Idempotent Coupling Fault



Stuck-at 0



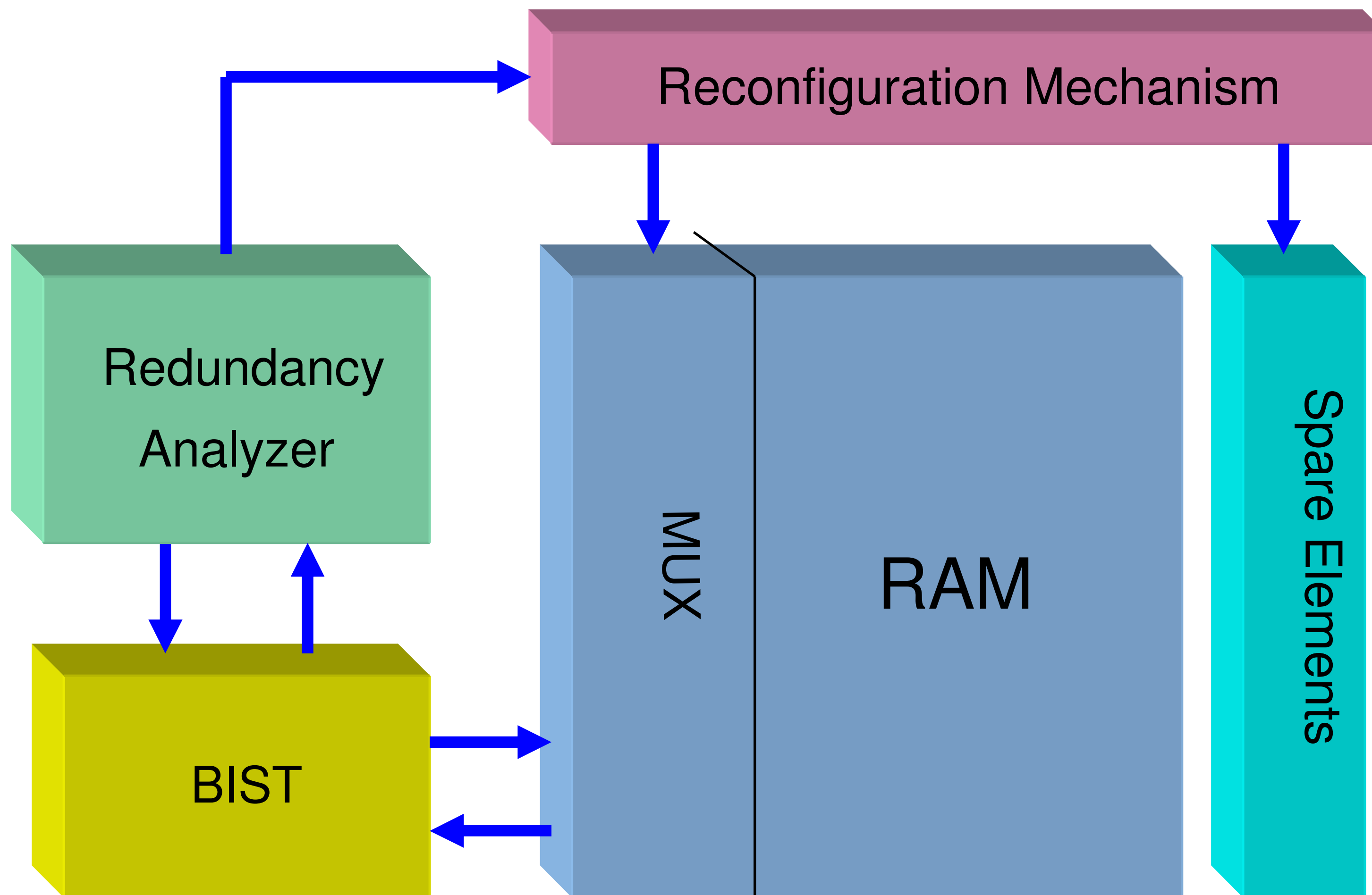
Redundancy and Repair

- Problem:
  - We keep shrinking RAM cell size and increasing RAM density and capacity. How do we maintain the yield?
- Solutions:
  - Fabrication
    - Material, process, equipment, etc.
  - Design
    - Device, circuit, etc.
  - Redundancy and repair
    - On-line
      - EDAC (extended Hamming code; product code)
    - Off-line
      - Spare rows, columns, blocks, etc.

From BIST to BISR

- BIST: built-in self-test
- BIECA: built-in error catch & analysis
  - BISD: built-in self diagnosis
  - BIRA: built-in redundancy analysis
- BISR: built-in self-repair

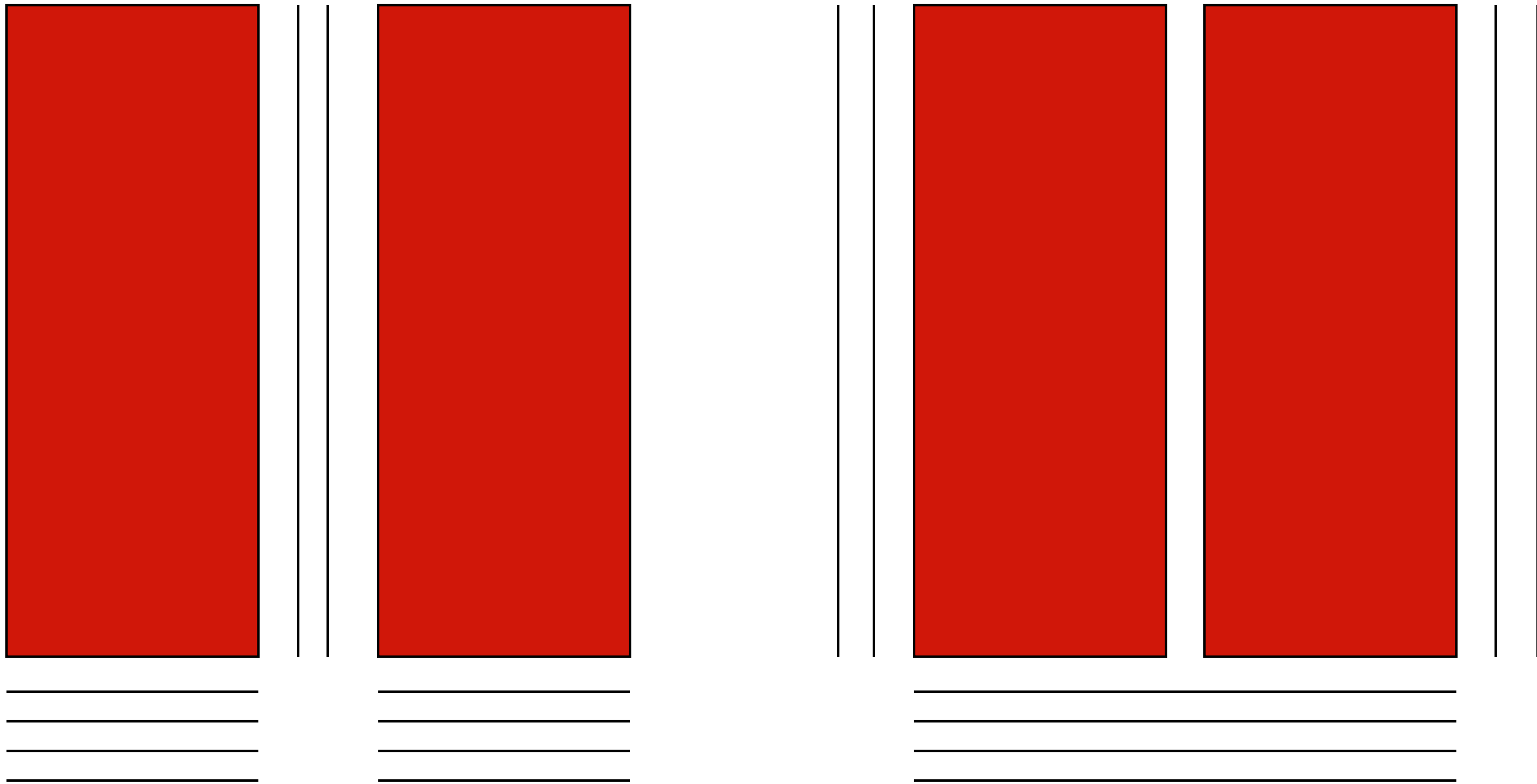
RAM Built-In Self-Repair (BISR)



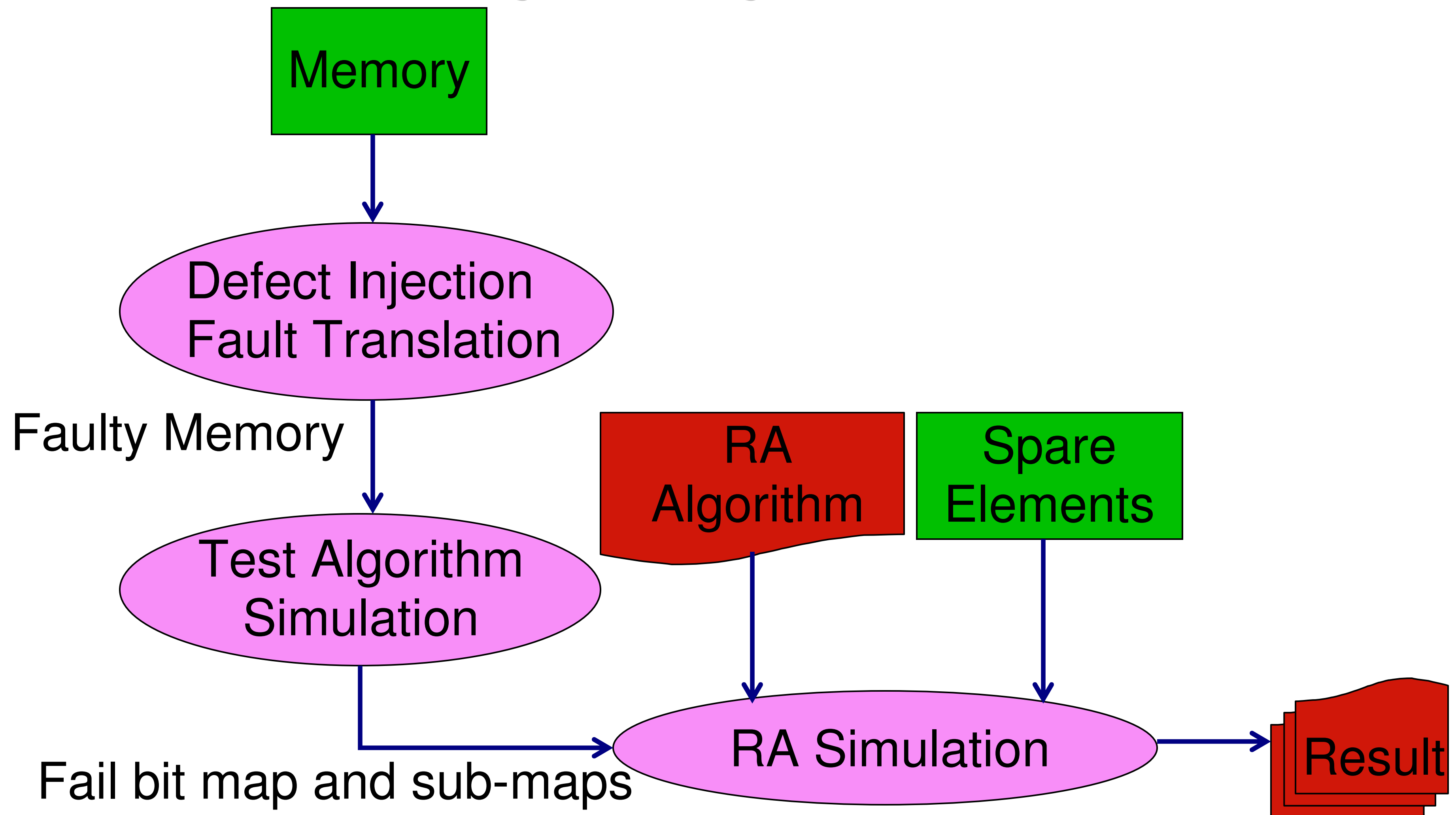
RAM Redundancy Allocation

- 1-D: spare rows (or columns) only
  - SRAM
  - Algorithm: Must-Repair
- 2-D: spare rows and columns (or blocks)
  - Local and/or global spares
  - NP-complete problem
  - Conventional algorithm:
    - Must-Repair phase
    - Final-Repair phase
      - Repair-Most (greedy) [Tarr *et al.*, 1984]
      - Fault-Driven (exhaustive, slow) [Day, 1985]
      - Fault-Line Covering (b&b) [Huang *et al.*, 1990]

Redundancy Architectures



Redundancy Analysis Simulation

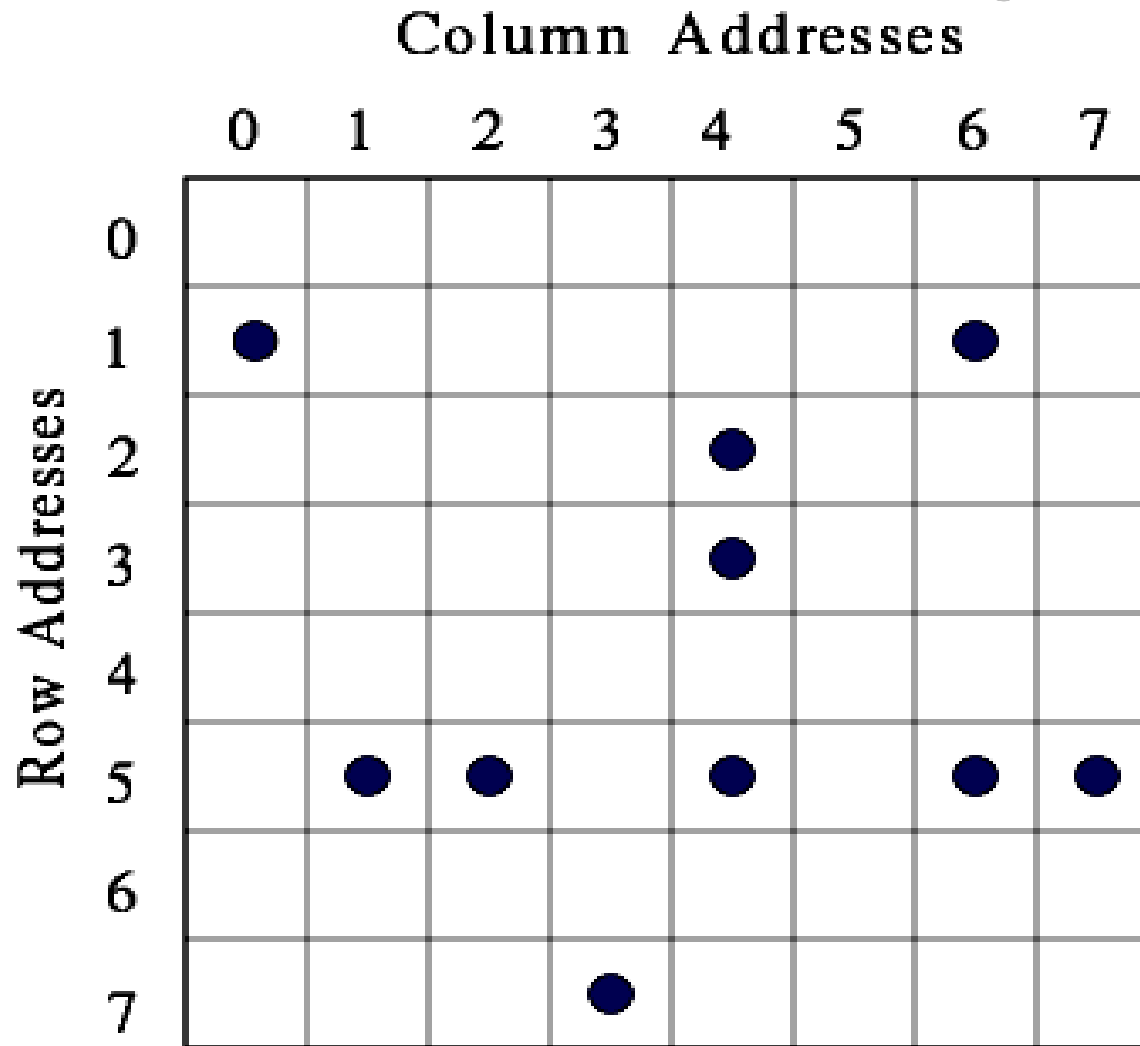


Ref: MTDT02

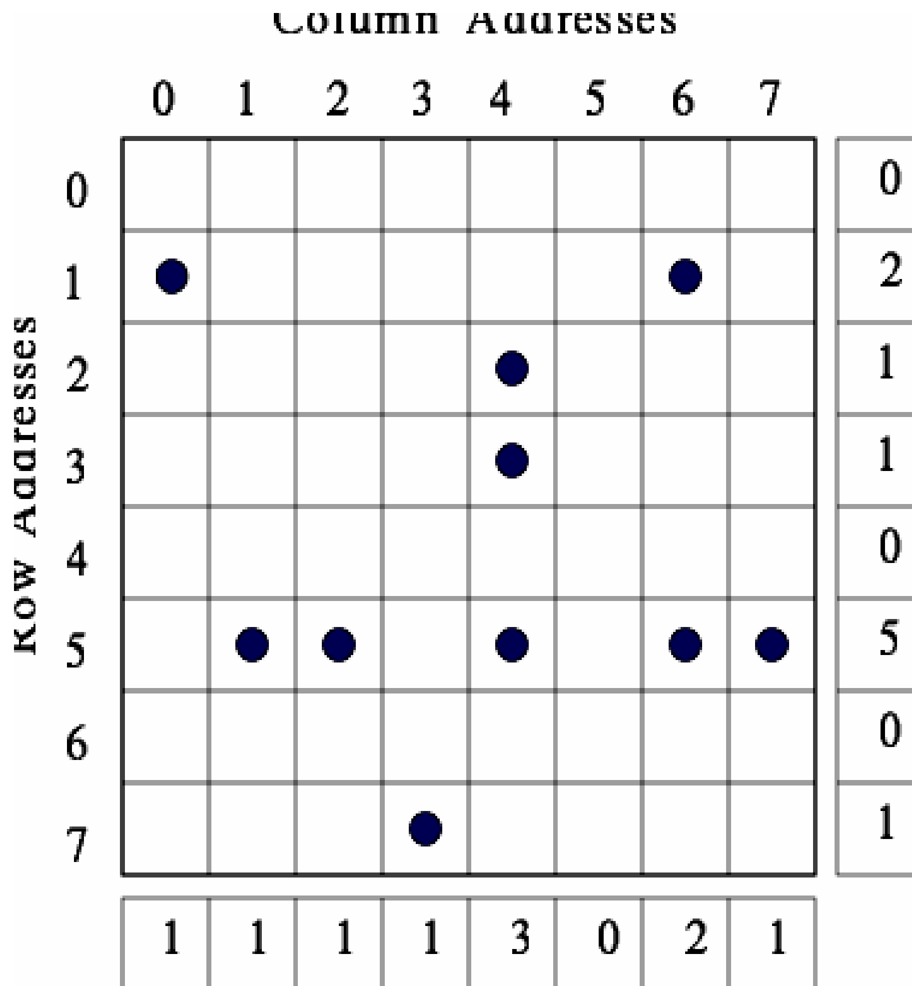
Definitions

- **Faulty line**: row or column with at least one faulty cell
  - A faulty line is **covered** if all faulty cells in the line are repaired by spare rows and/or columns.
- A faulty cell not sharing any row or column with any other faulty cell is an **orthogonal faulty cell**
- *r*: number of (available) spare rows
- *c*: number of (available) spare columns
- *F*: number of faulty cells in a block
- *F'*:number of orthogonal faulty cells in a block

Example Block with Faulty Cells

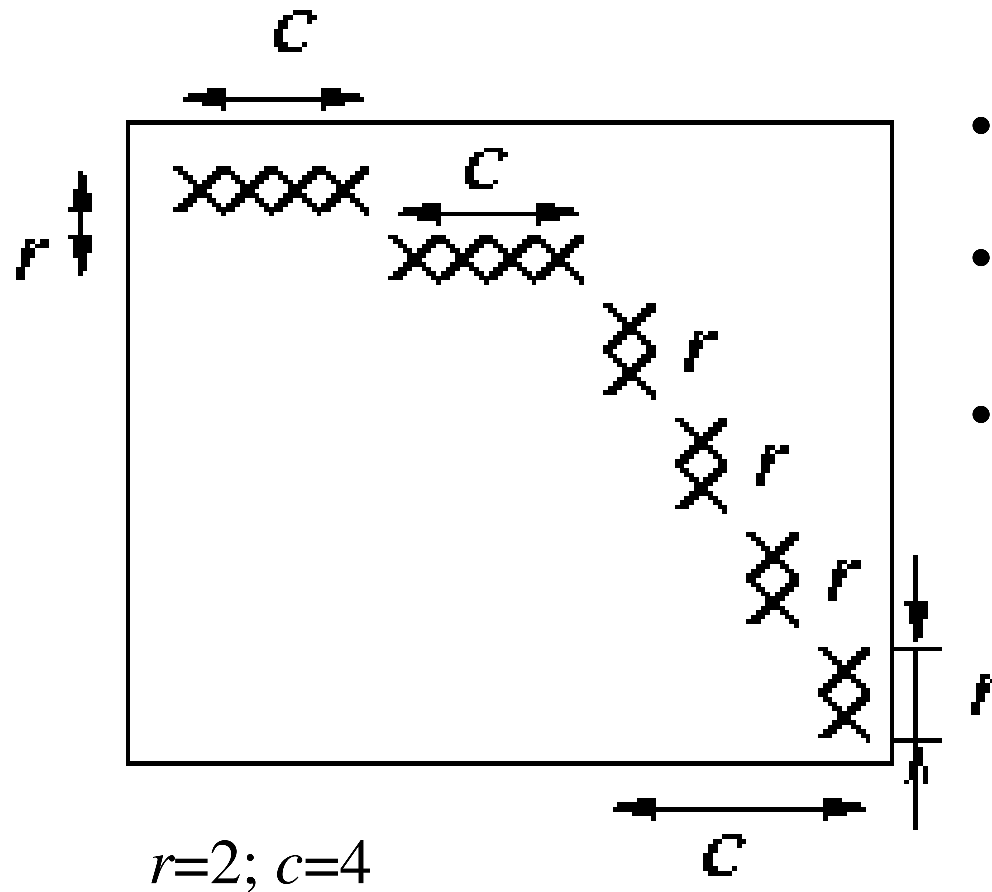


Repair-Most (RM)



1. **Run BIST and construct bitmap**
2. **Construct row and column error counters**
3. **Run Must-Repair algorithm**
4. **Run greedy final-repair algorithm**

Worst-Case Bitmap (After Must-Repair)



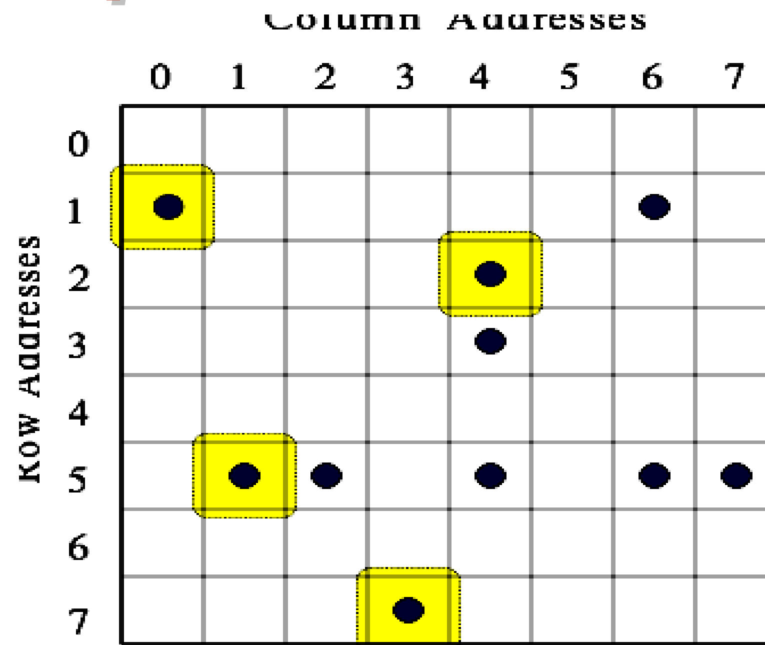
- **Max** $F=2rc$
- **Max** $F'=r+c$
- **Bitmap size:** $(rc+c)(cr+r)$

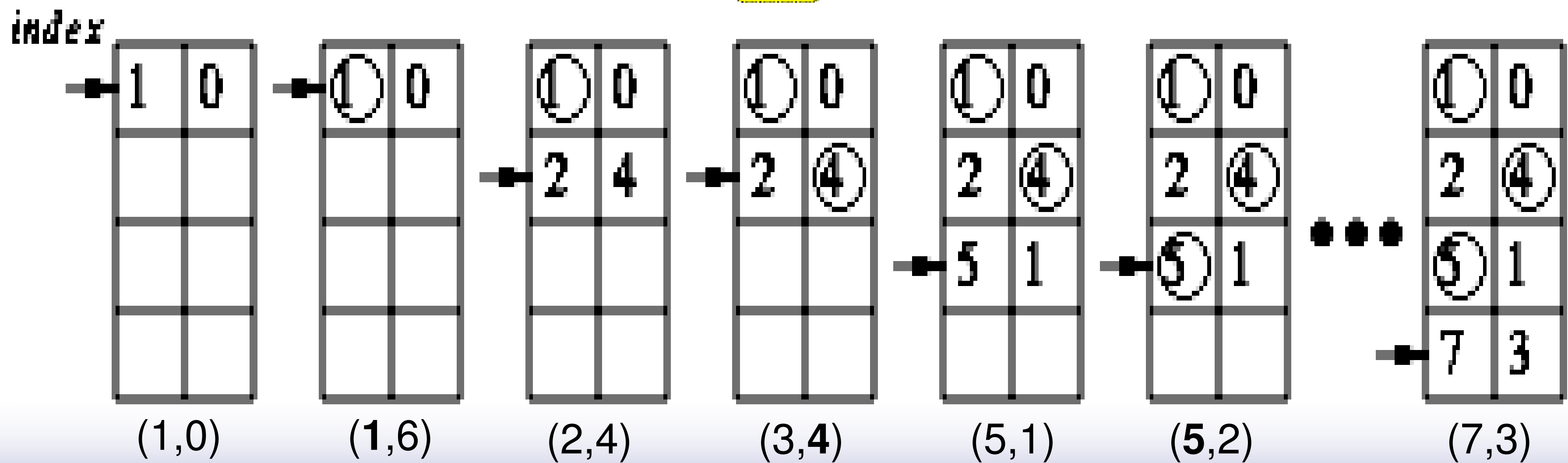
Essential Spare Pivoting (ESP)

- Maintain high repair rate without using a bitmap
  - Small area overhead
- Fault Collection (FC)
  - Collect and store faulty-cell address using row-pivot and column-pivot registers
    - If there is a match for row (col) pivot, the pivot is an essential pivot
    - If there is no match, store the row/col addresses in the pivot registers
  - If $F > r+c$, the RAM is irreparable
- Spare Allocation (SA)
  - Use row and column pivots for spare allocation
    - Spare rows (cols) for essential row (col) pivots
  - SA for orthogonal faults

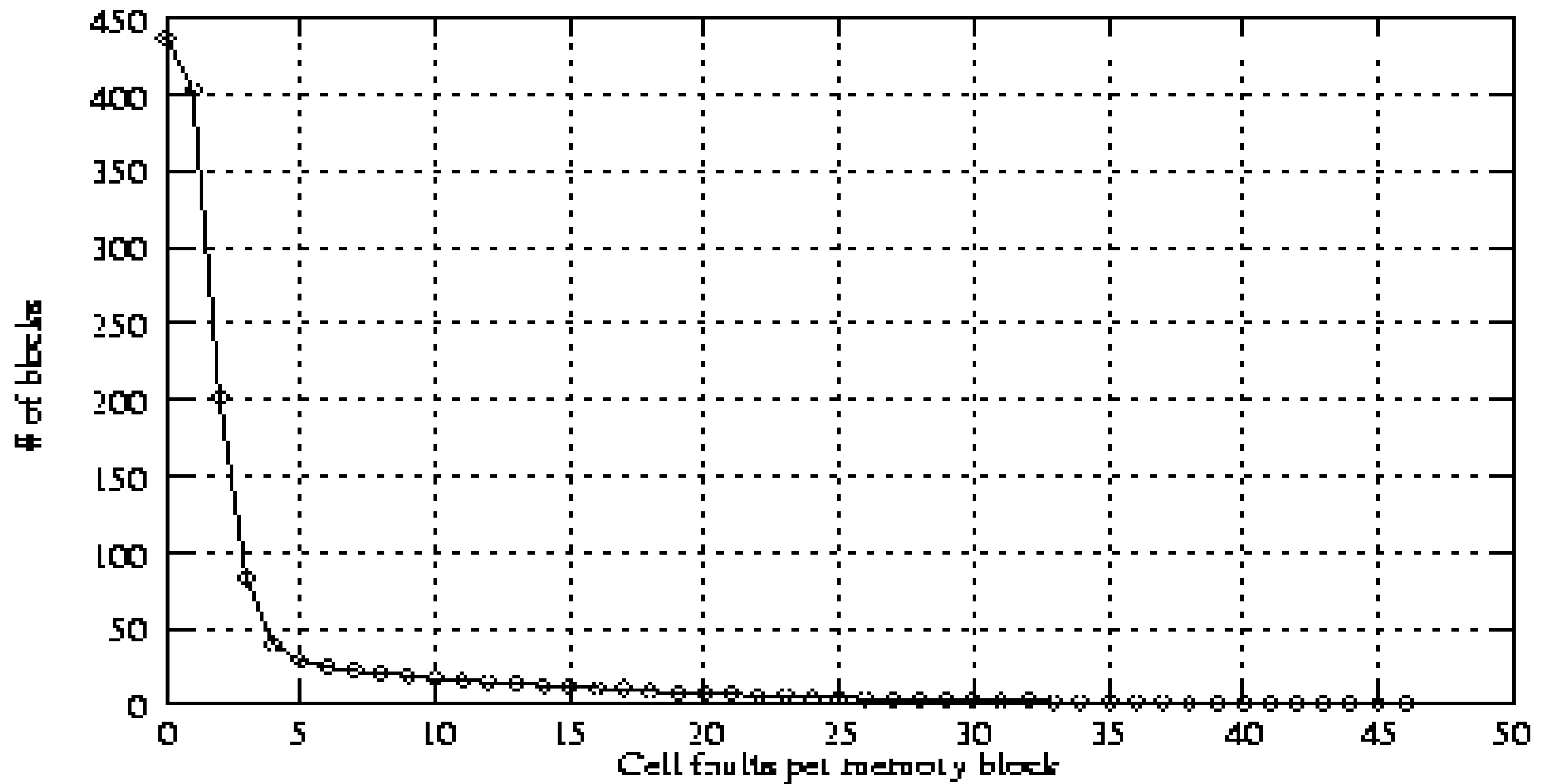
Ref: Huang *et al.*, IEEE TR, 11/03

ESP Example



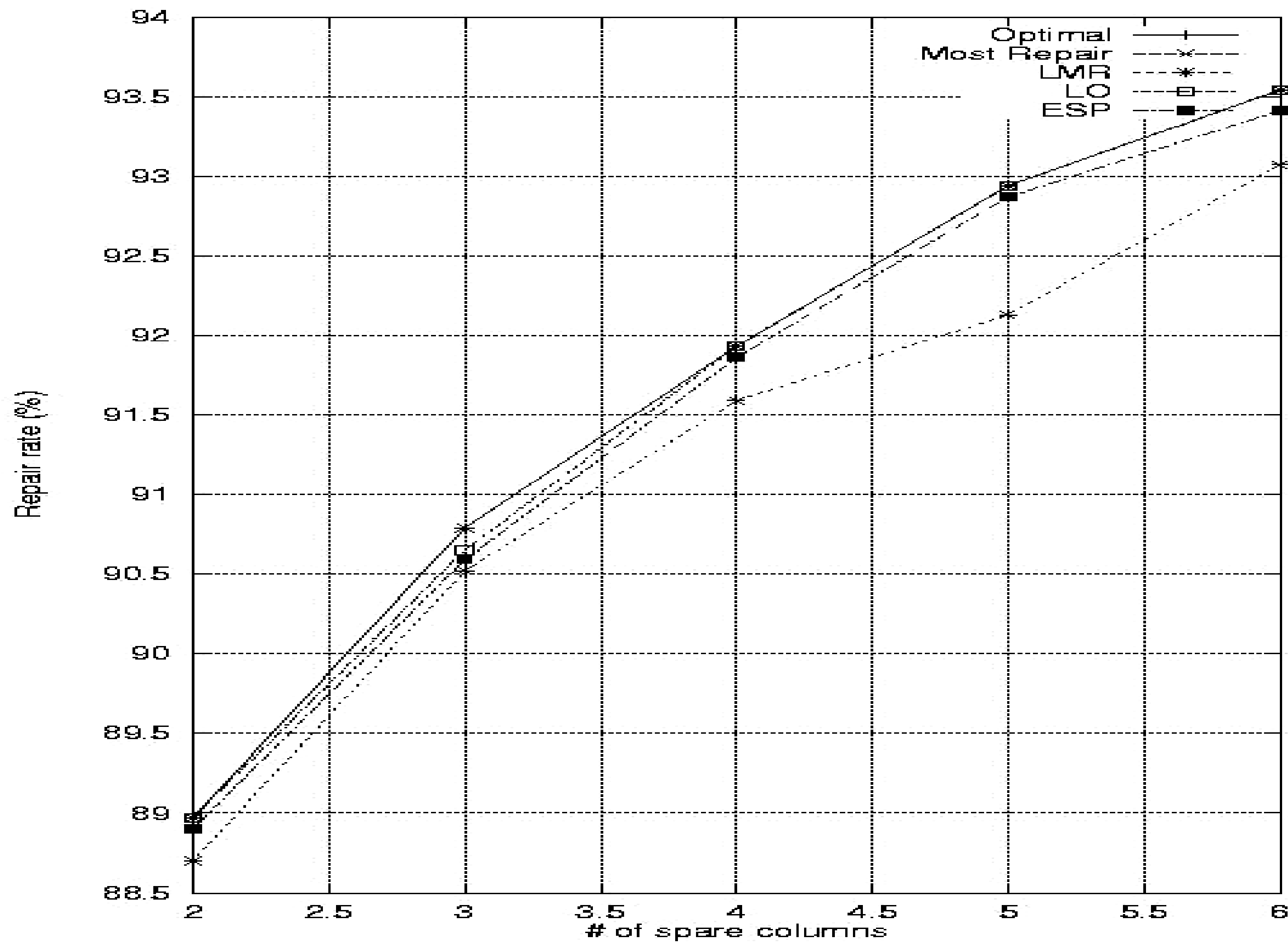


Cell Fault Size Distribution

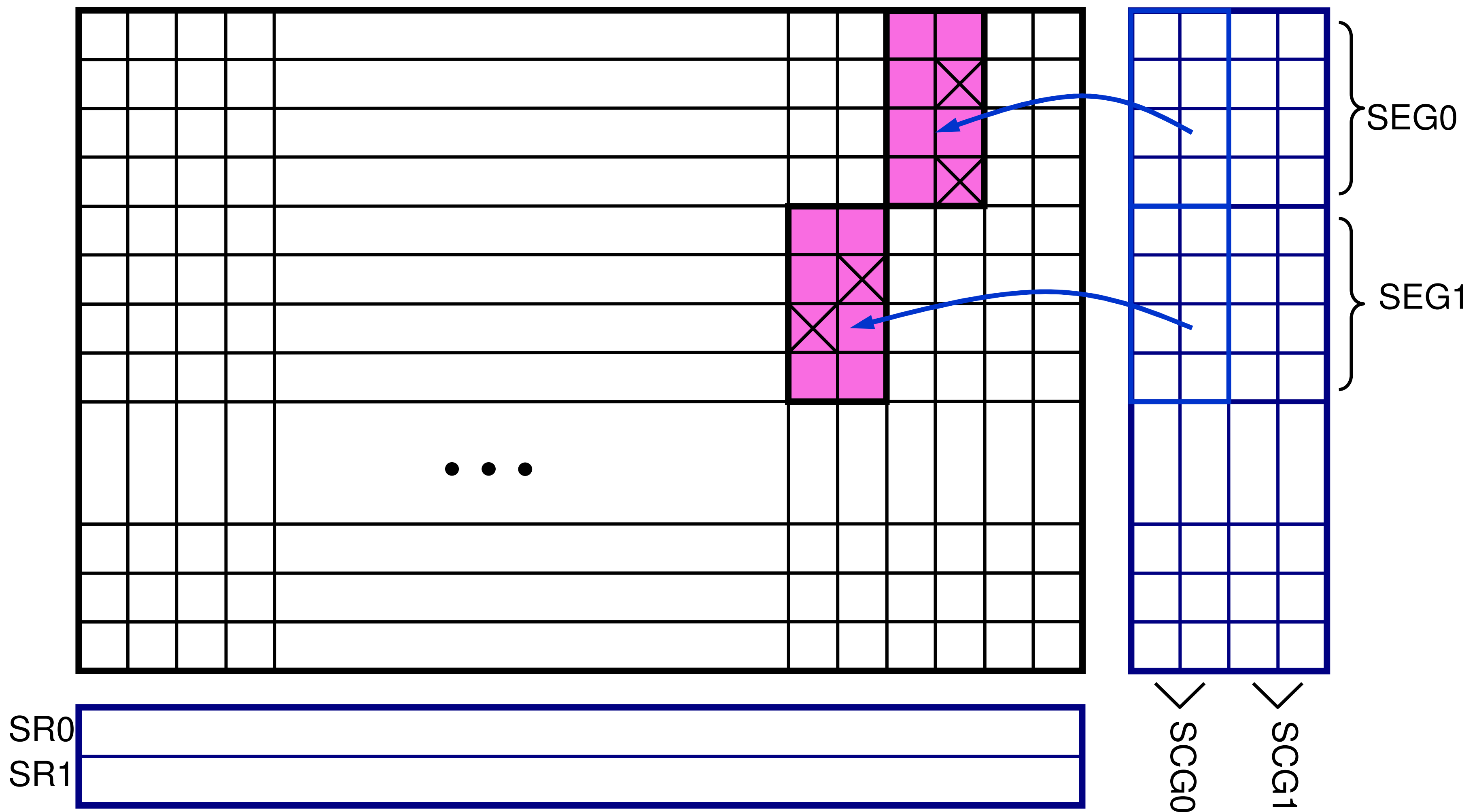


Mixed Poisson-exponential distribution

Repair Rate (r=10)



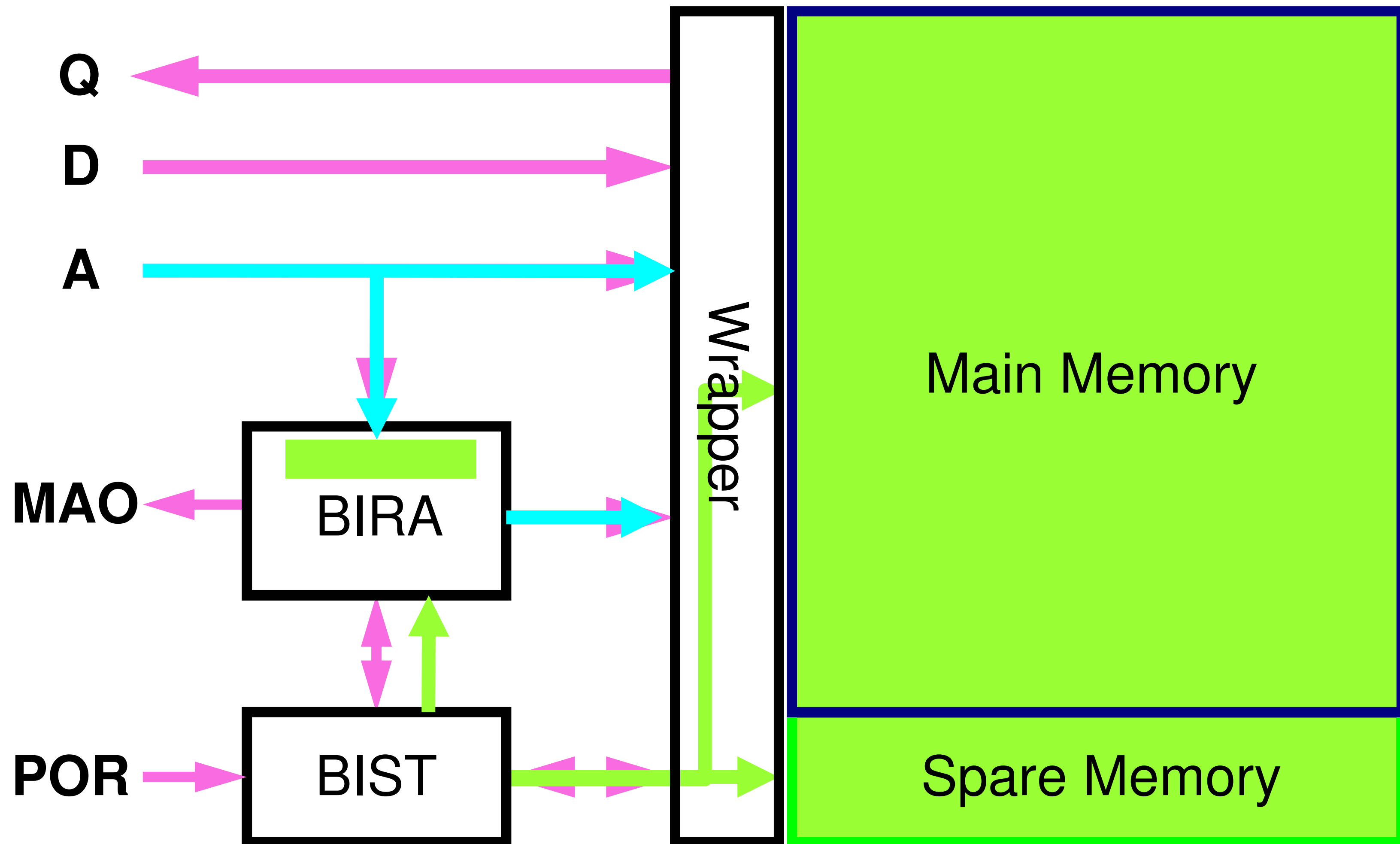
Redundancy Organization



SR: Spare Row; SCG: Spare Column Group; SEG: Segment

ITC03

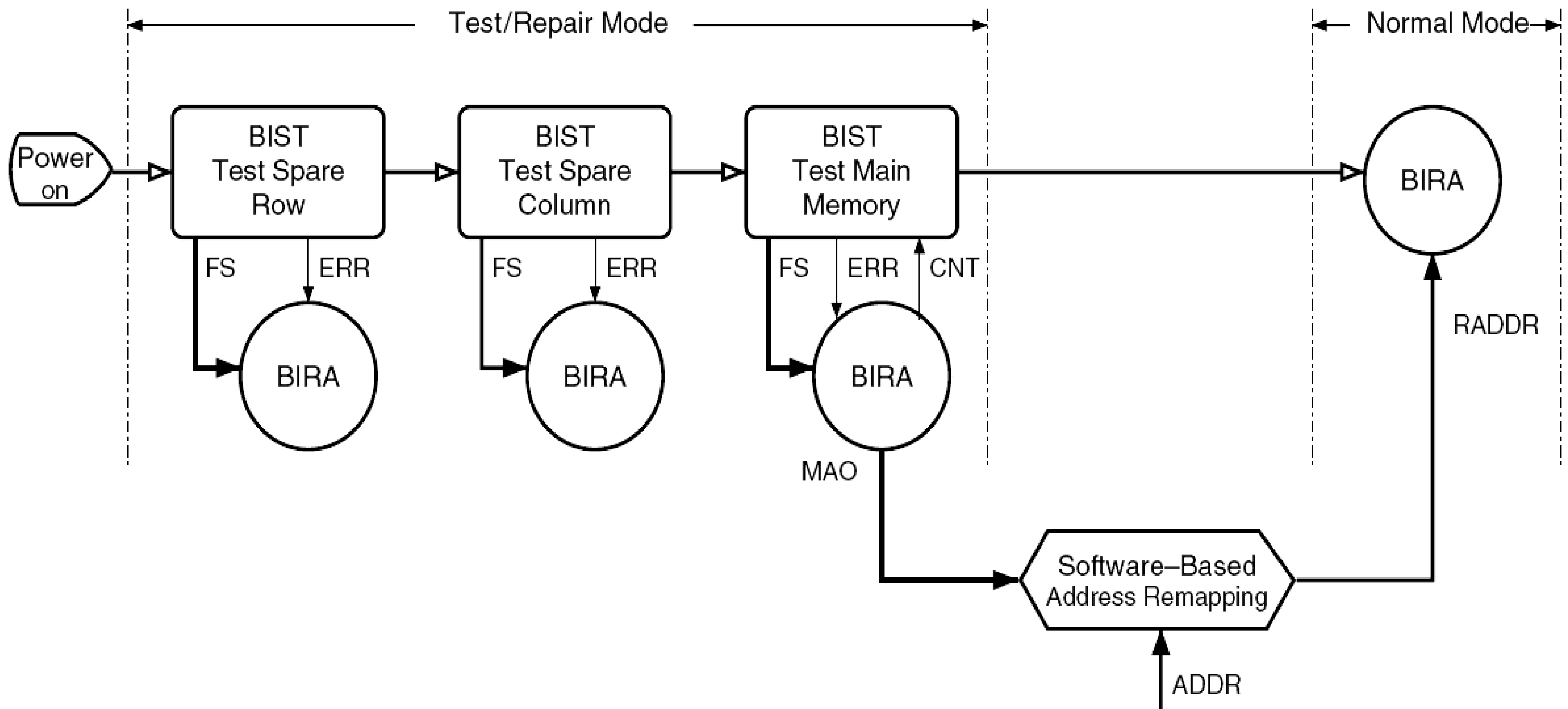
BISR Architecture



MAO: mask address output; POR: power-on reset

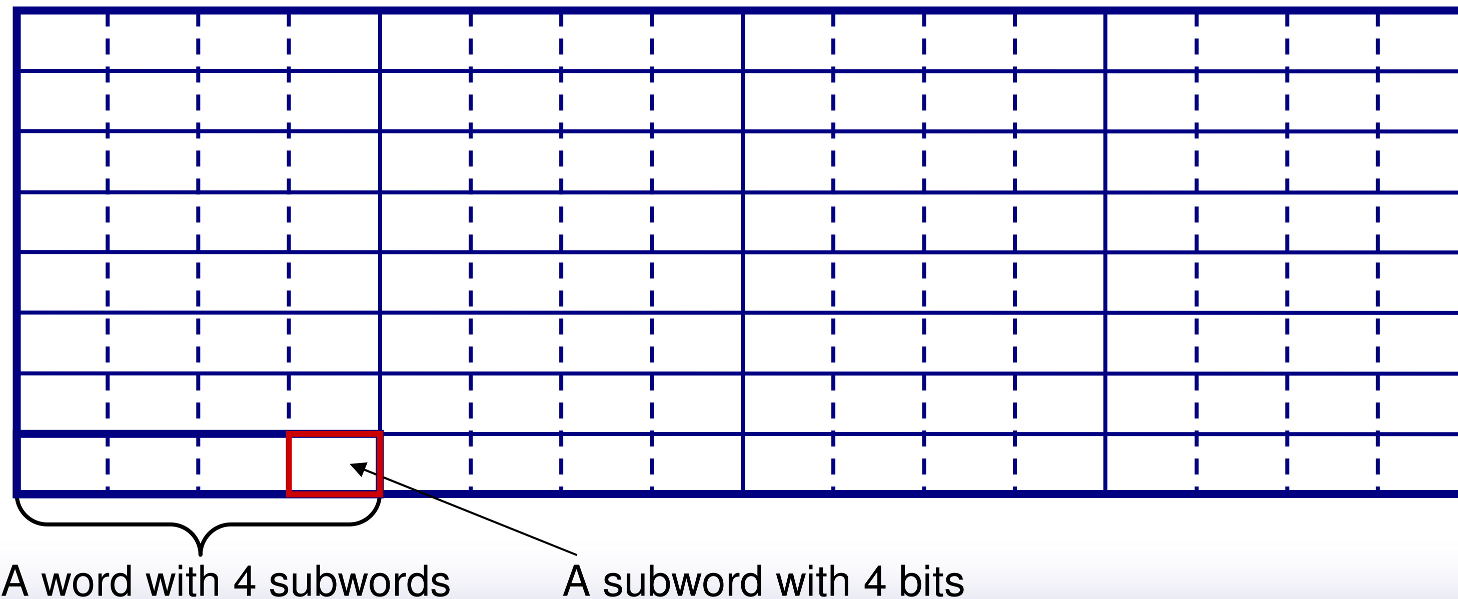
Ref: ITC03

Power-On BISR Procedure



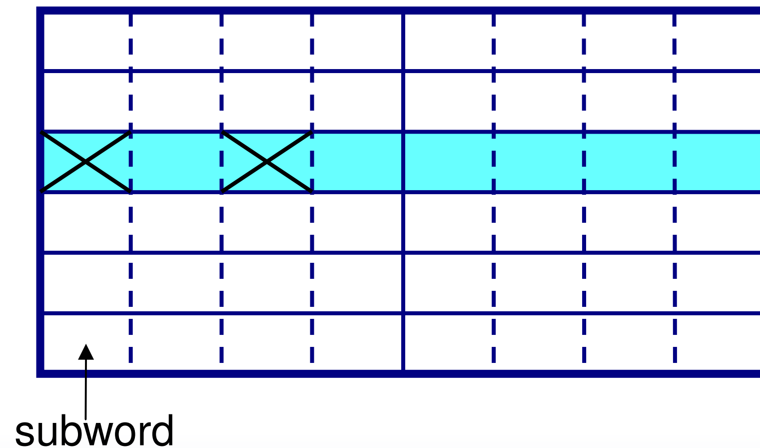
Subword Definition

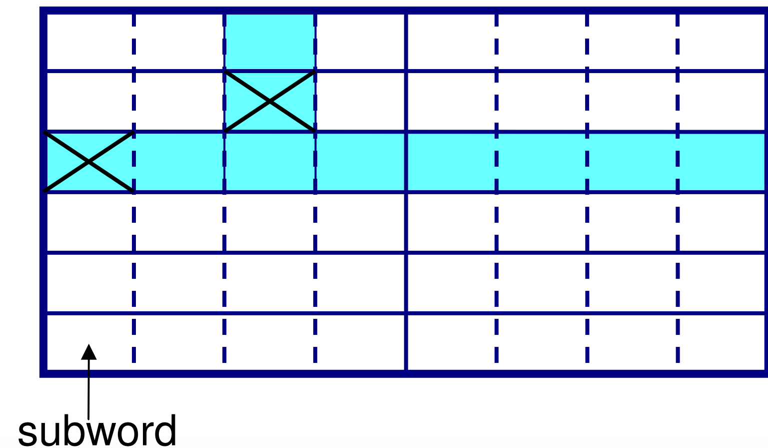
- Subword
  - A subword is consecutive bits of a word.
  - Its length is the same as the group size.
- Example: a 32x16 RAM with 3-bit row address and 2-bit column address



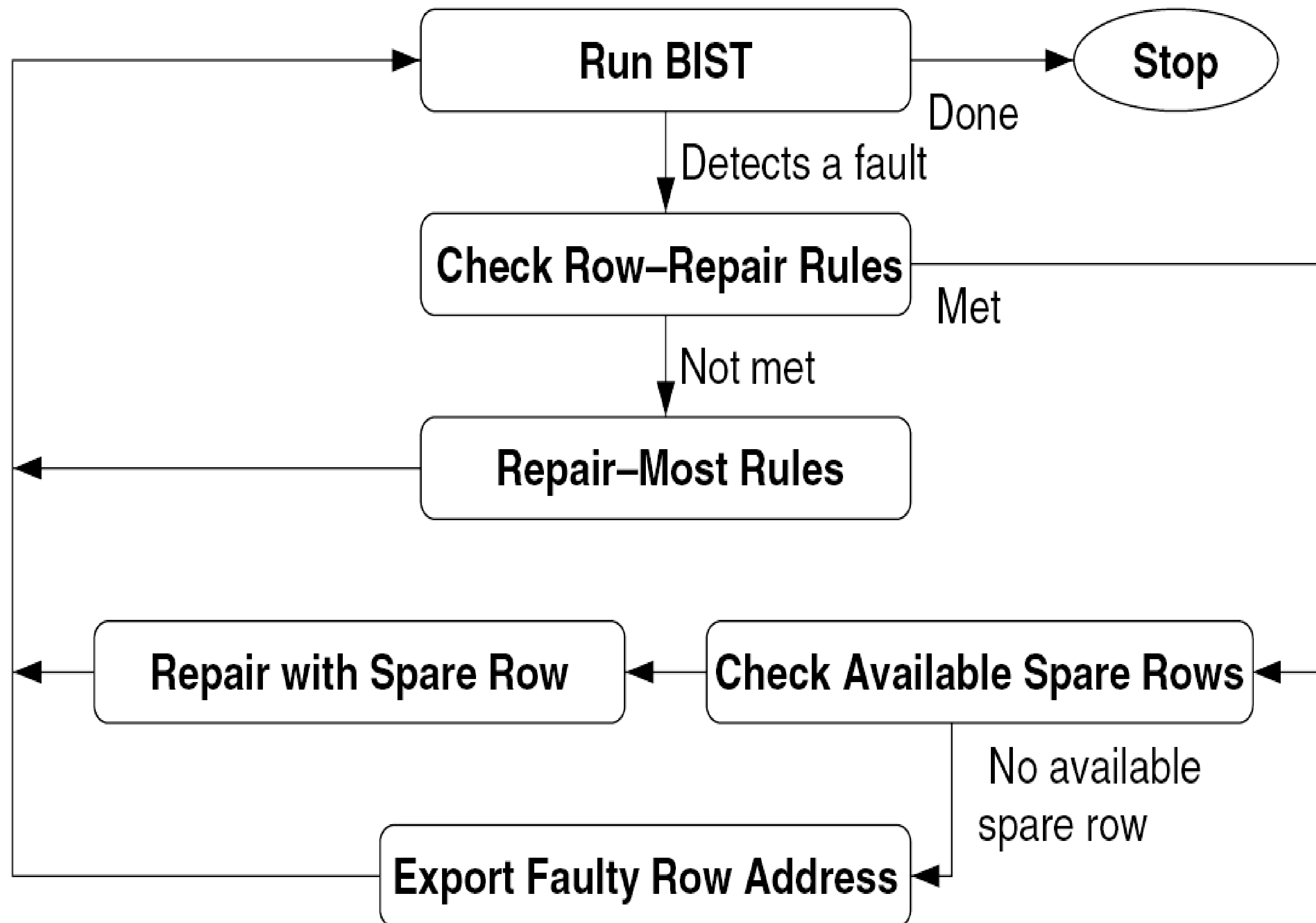
Row-Repair Rules

- To reduce the complexity, we use two row-repair rules
  - If a row has multiple faulty, we repair the faulty row by a spare row if available.
  - If there are multiple faulty subwords with the same column address and different row addresses within a segment, the last detected faulty subword should be repaired with an available spare row.
- Examples:

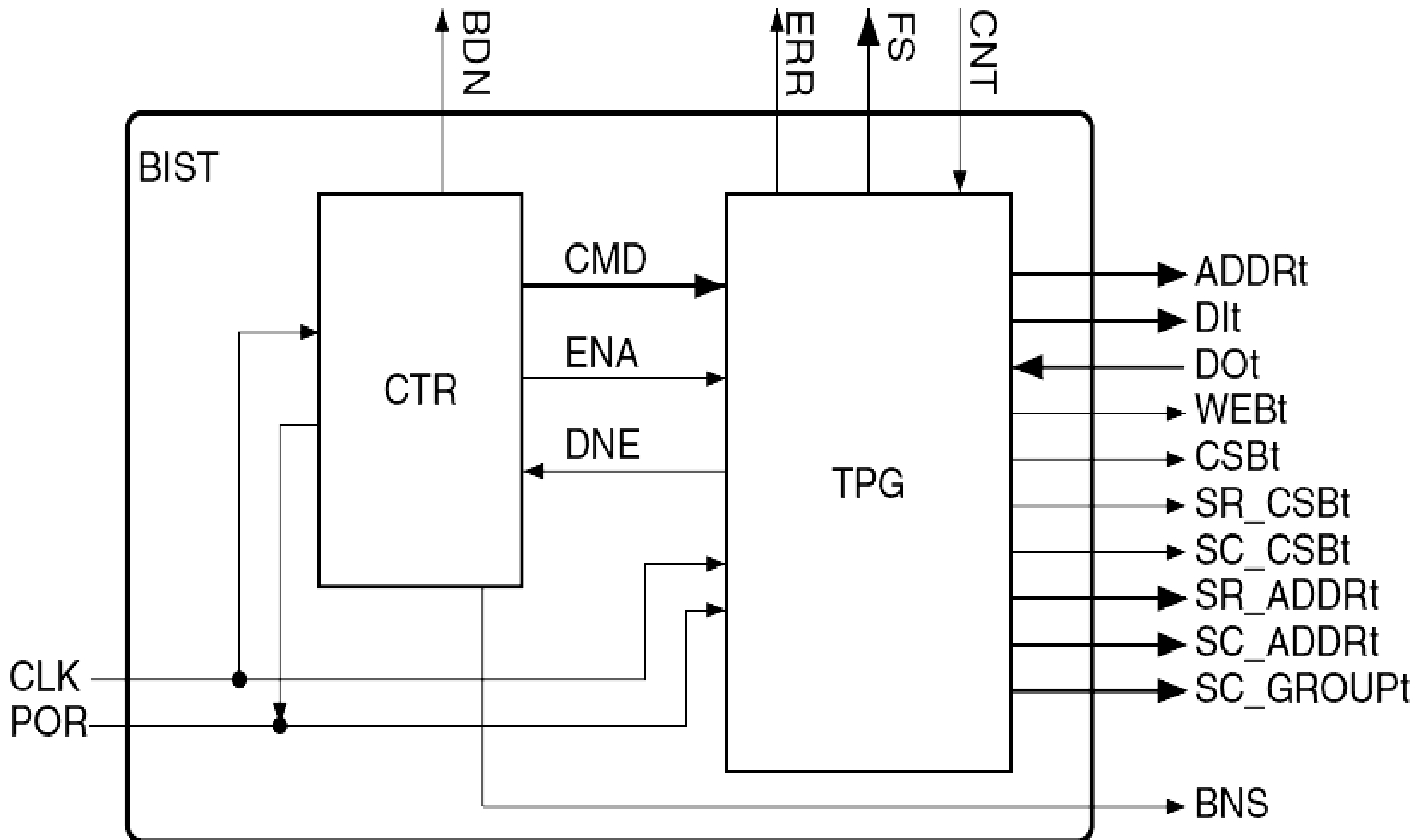




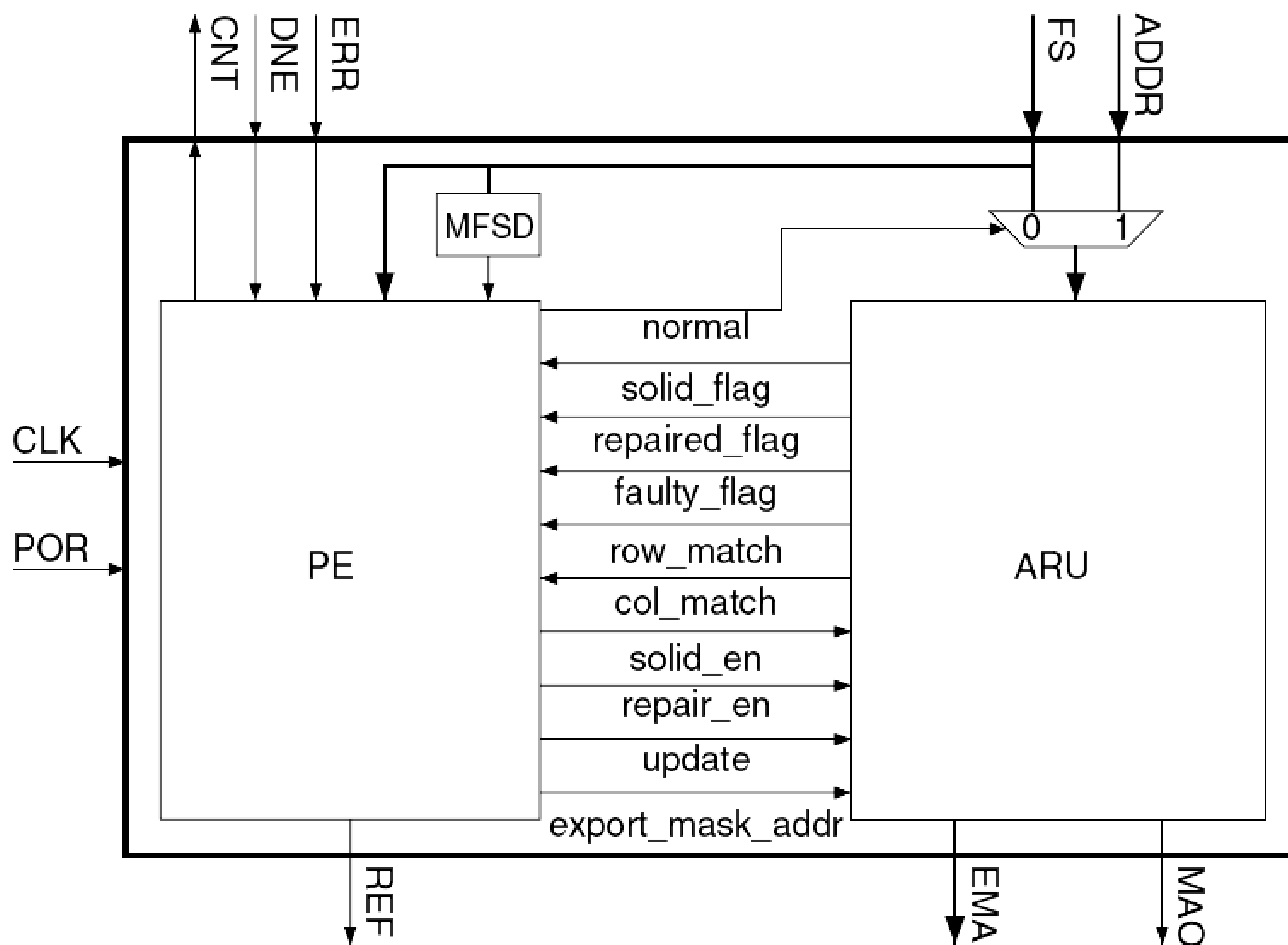
BIRA Procedure



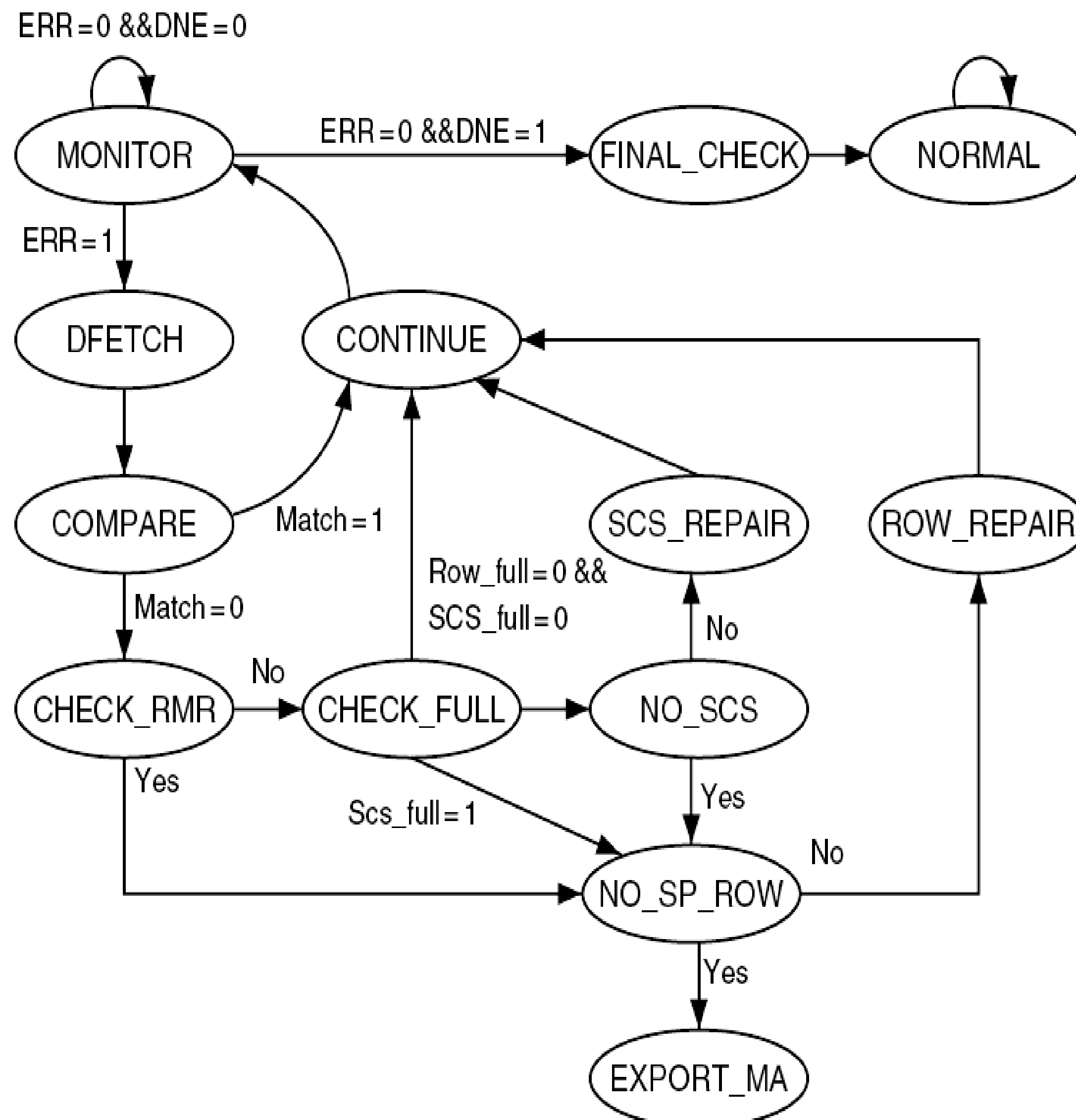
Basic BIST Module



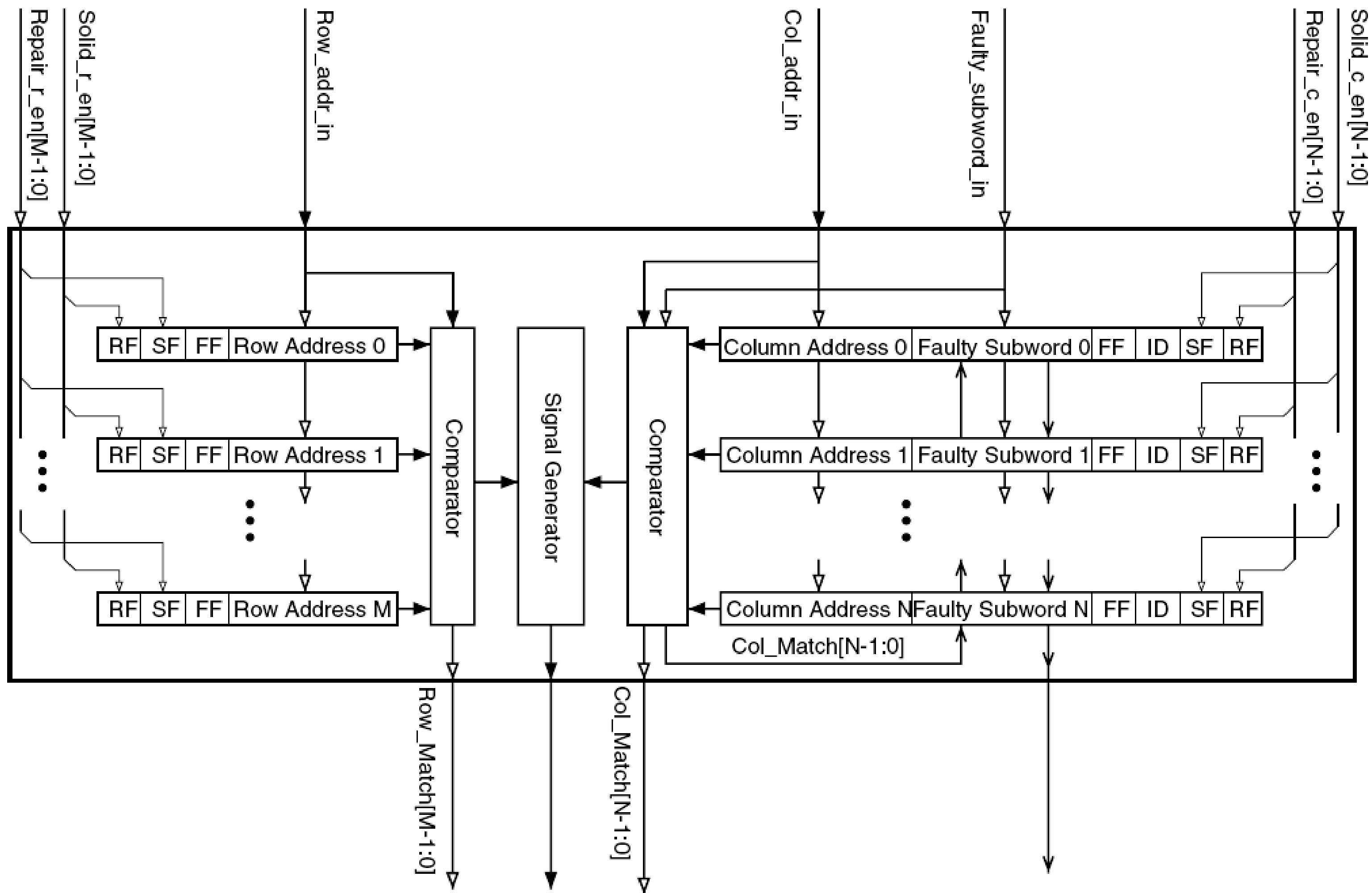
BIRA Module



State Diagram of PE



Block Diagram of ARU

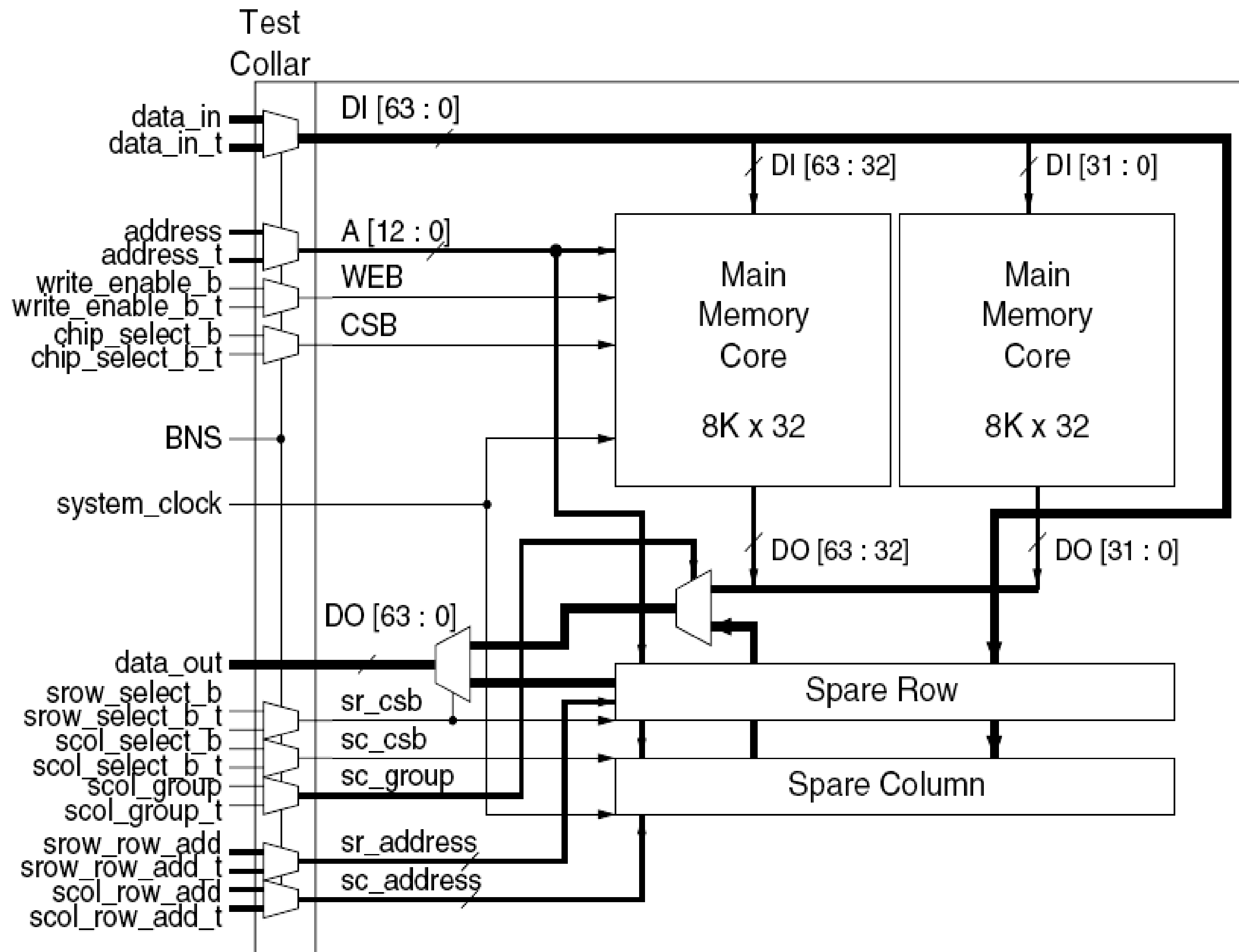


Repair Rate Analysis

- Repair rate
  - The ratio of the number of repaired memories to the number of defective memories
- A simulator has been implemented to estimate the repair rate of the proposed BISR scheme

  [Huang *et al*., MTDT02]
- Industrial case:
  - SRAM size: 8Kx64
  - # of injected random faults: 1~10
  - # of memory samples: 534
  - RA algorithms: proposed and exhaustive search algorithms

An Industrial Case

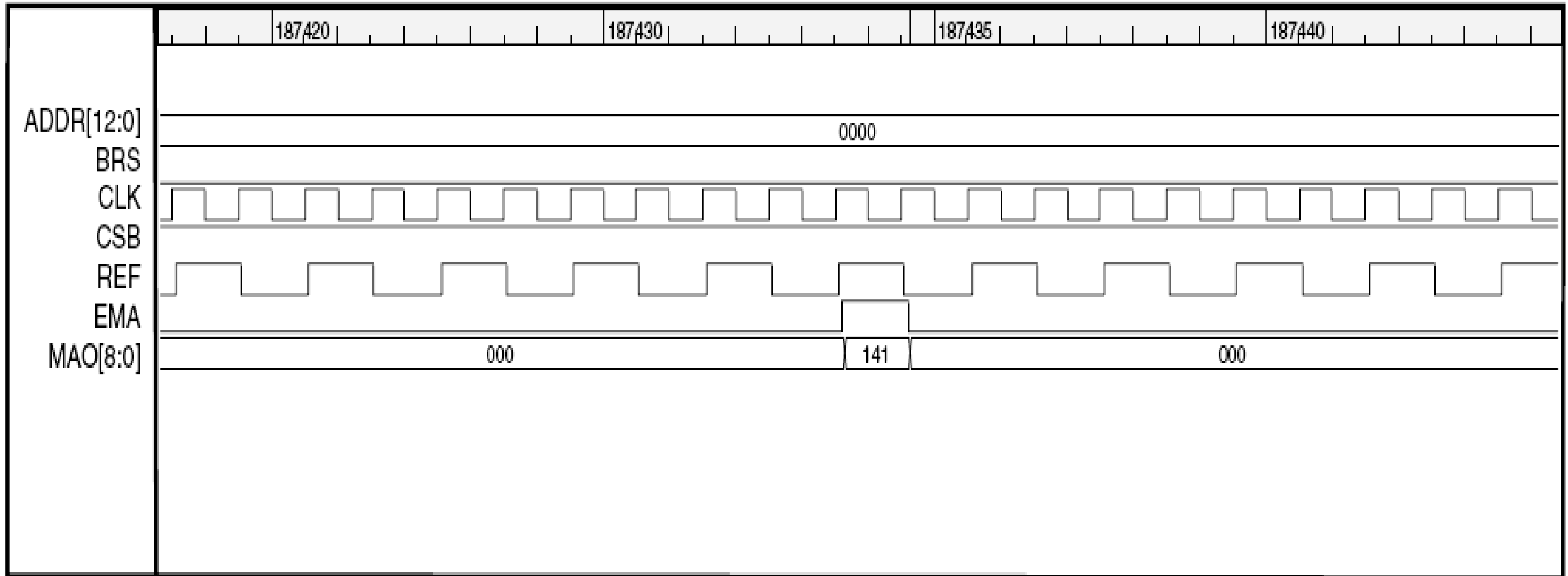


A 8Kx64 Repairable SRAM

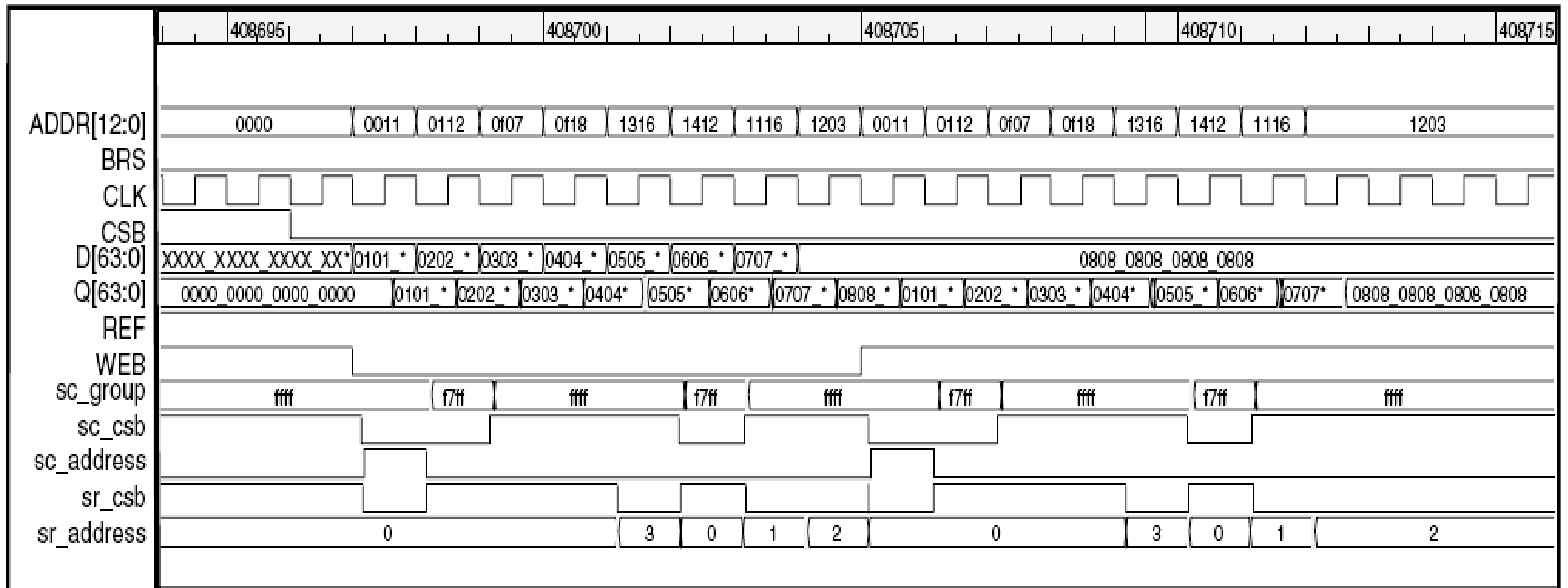
Technology: 0.25um
SRAM area: 6.5 mm^2
BISR area : 0.3 mm^2
Spare area : 0.3 mm^2
HO_{spare}: 4.6%
HO_{bisr}: 4.6%
Repair rate: 100% (if # random faults is no more than 10)

Redundancy: 4 spare rows and 2 spare column groups
Group size: 4

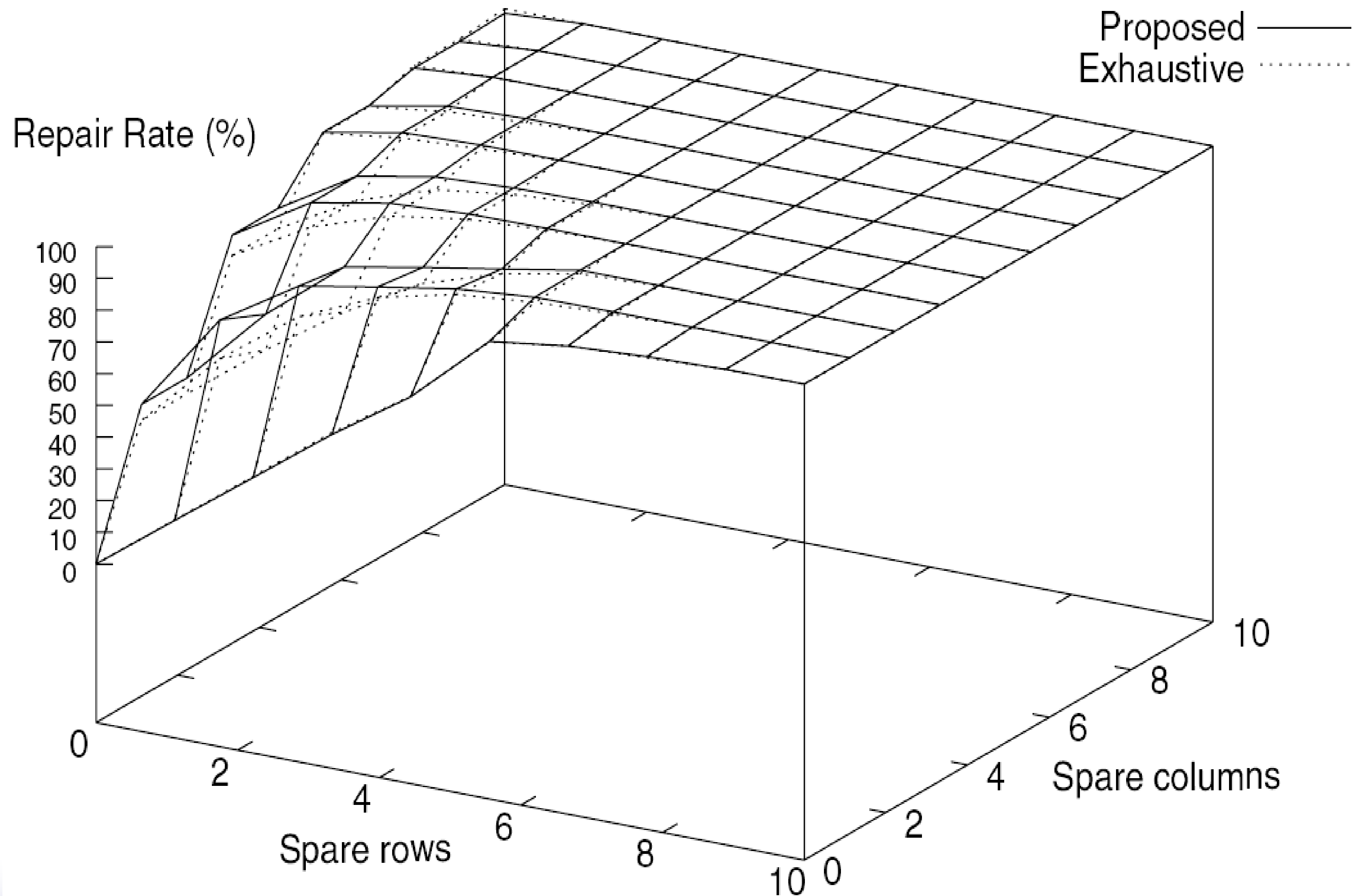
Waveform of EMA & MAO



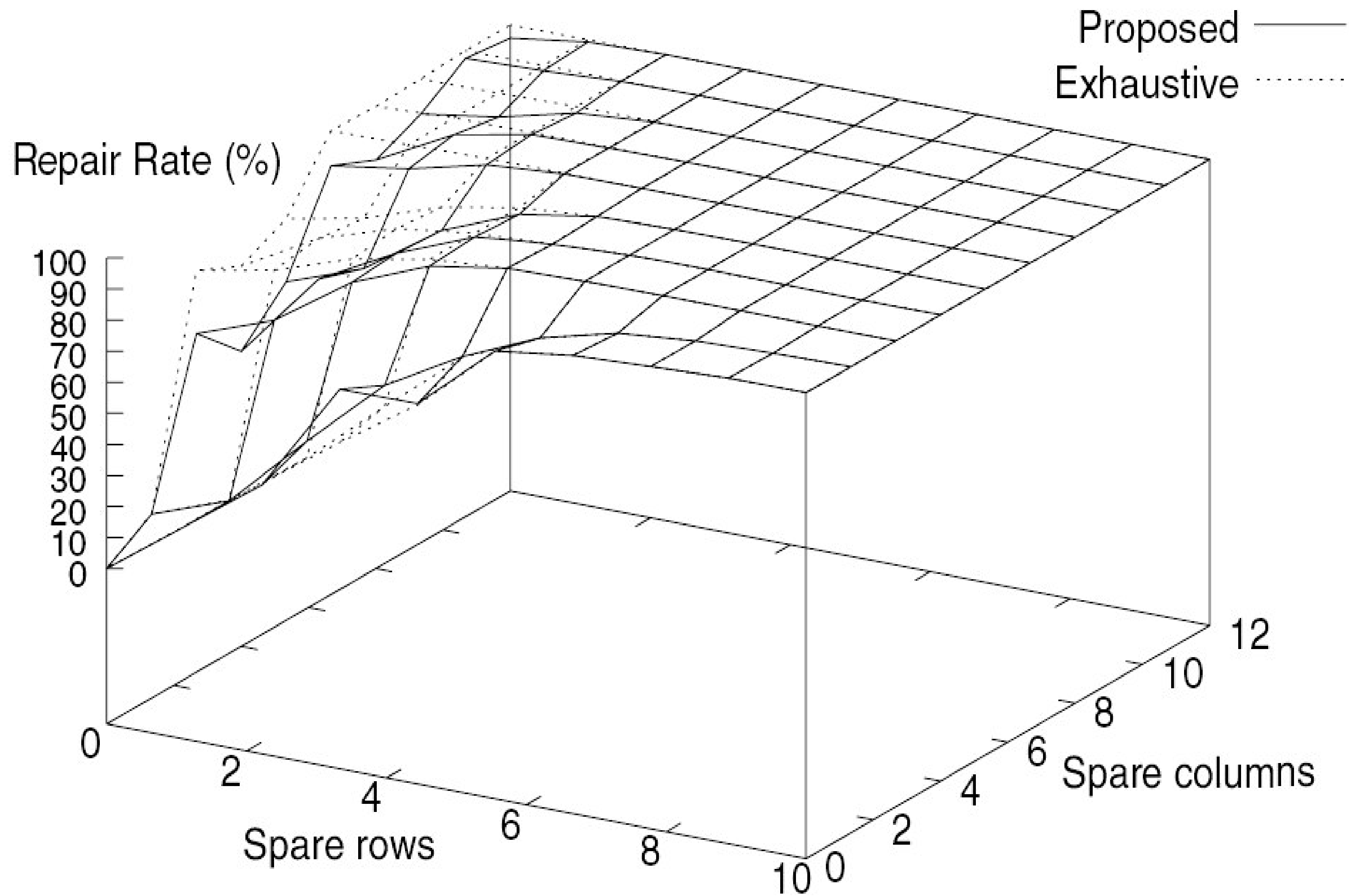
Normal Mode Waveform



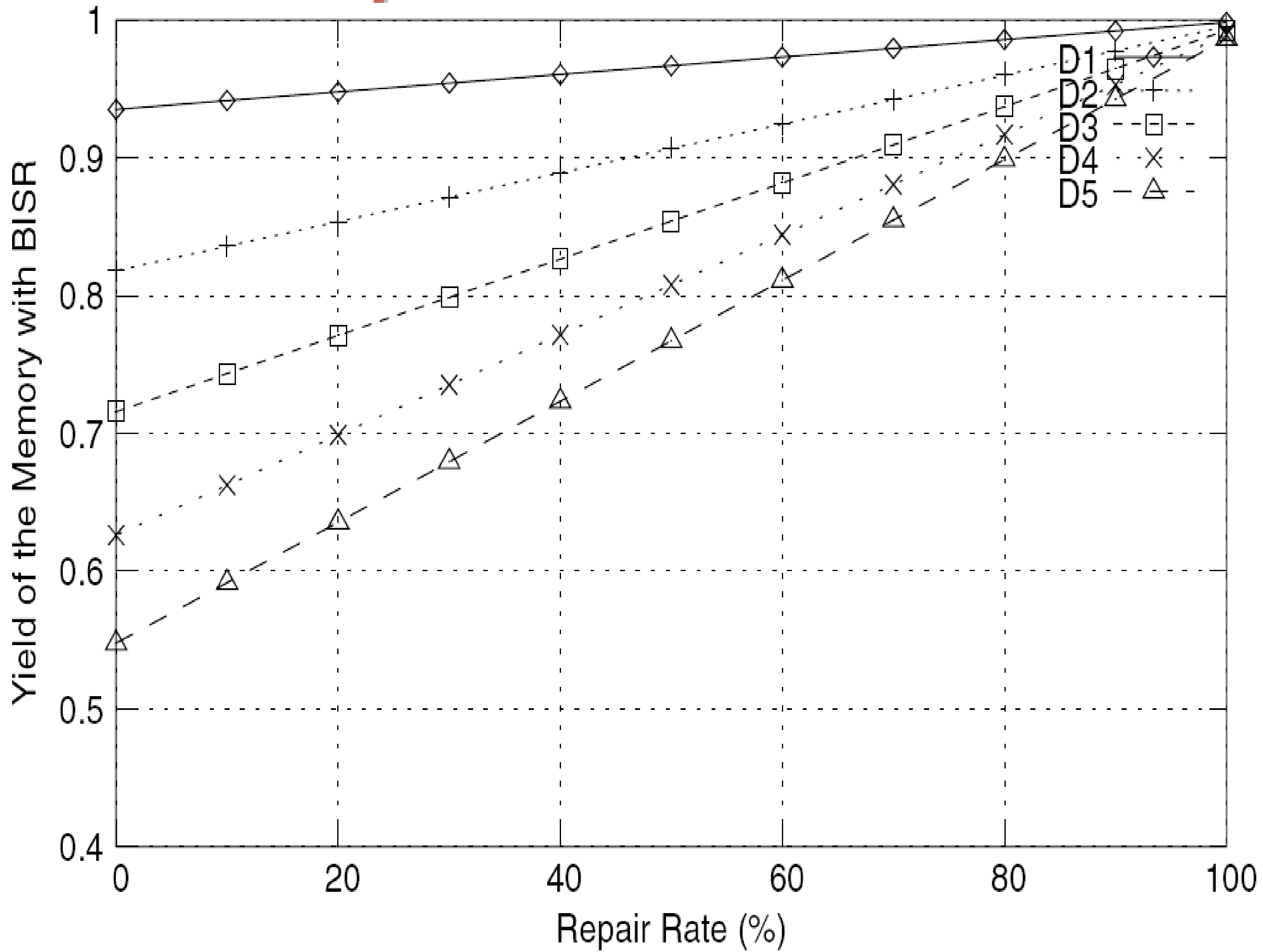
Repair Rate (Group Size 2)



Repair Rate (Group Size 4)



Yield vs. Repair Rate



Concluding Remarks

- BIST with diagnosis support
  - Fault type identification done by an offline diagnosis process using MECA
  - RAM design and process debugging for yield enhancement
- From BIST to BIRA
  - Effective implementation by ESP
    - Greedy algorithm
- An industrial case has been experimented
  - Full repair achieved (for # random faults no more than 10)
  - Only 4.6% area overhead for the 8Kx64 SRAM