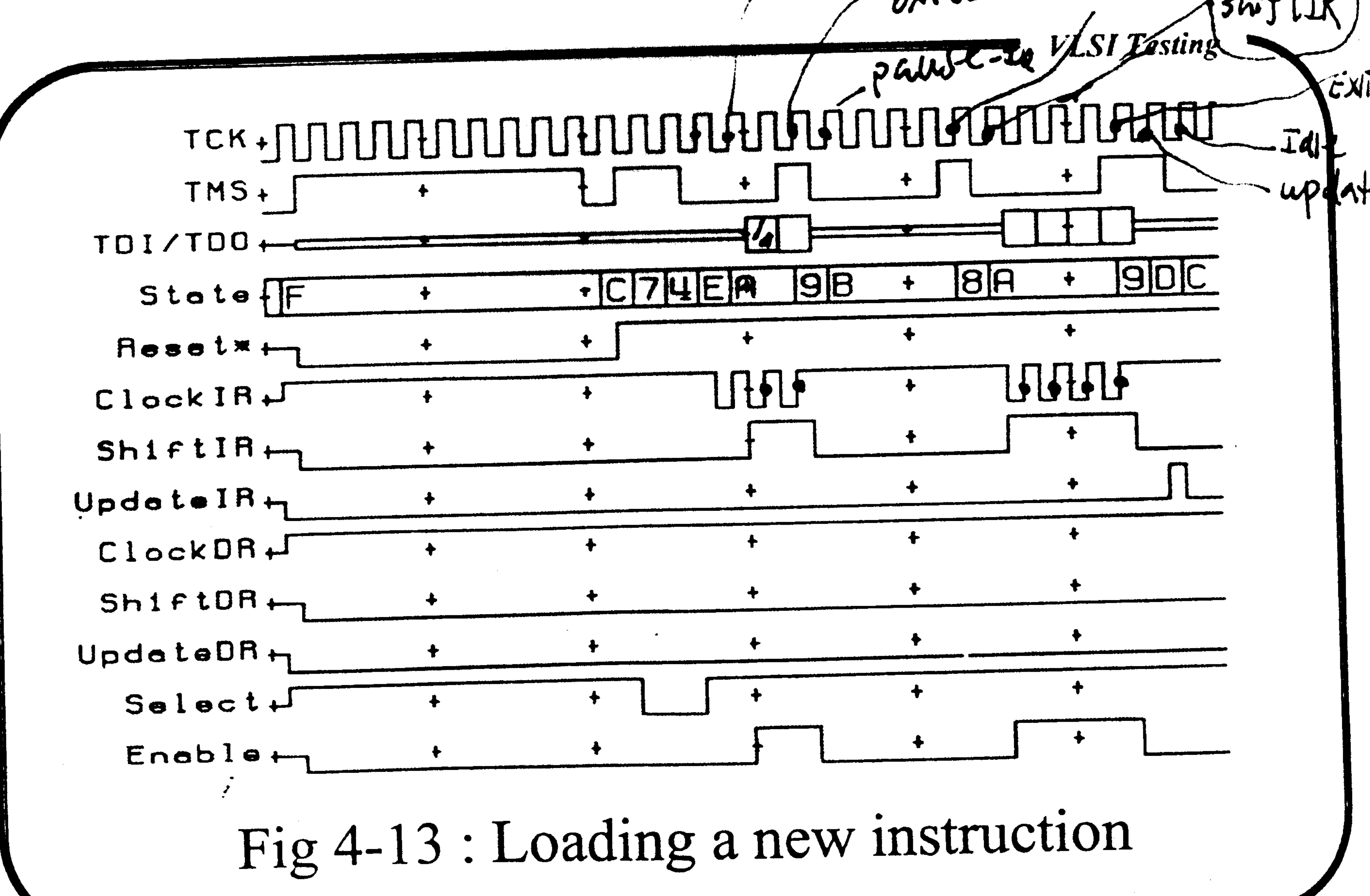
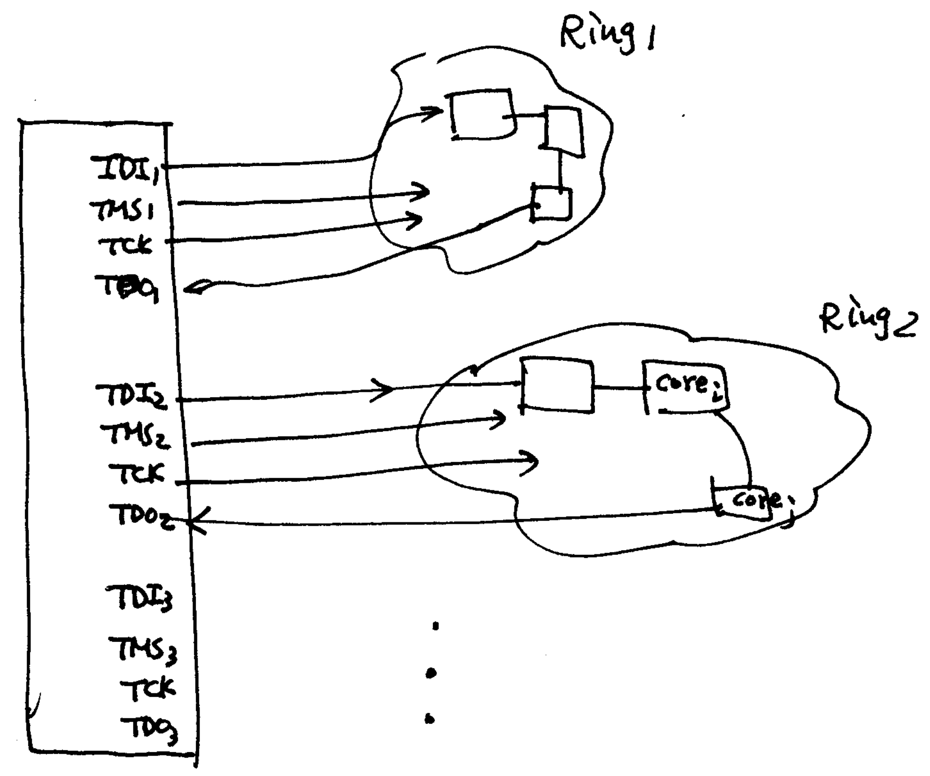


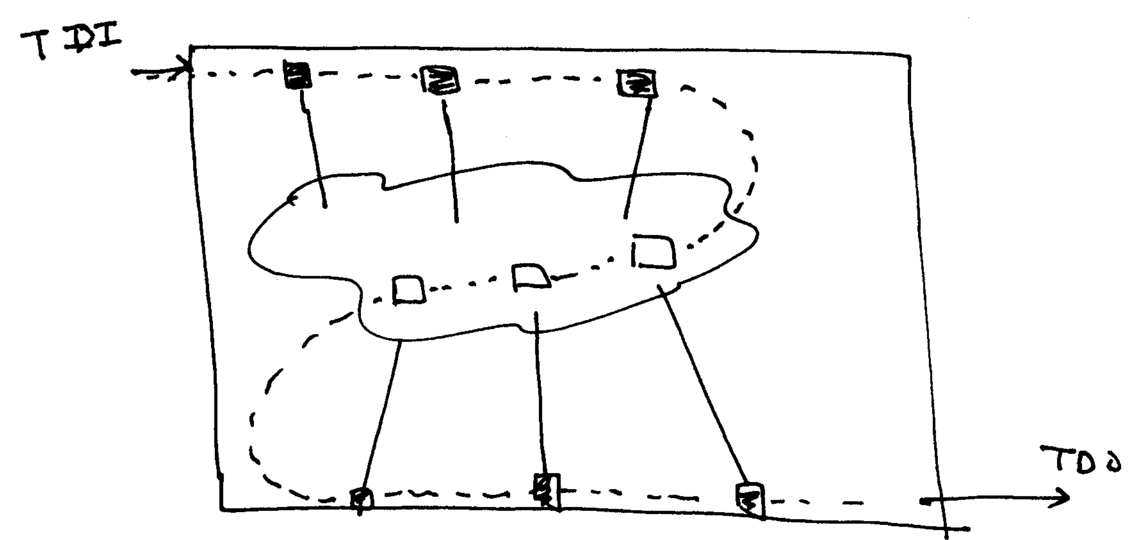
Fig 4-13 : Loading a new instruction

Fig 4-15 : Loading new test data

- Star architecture

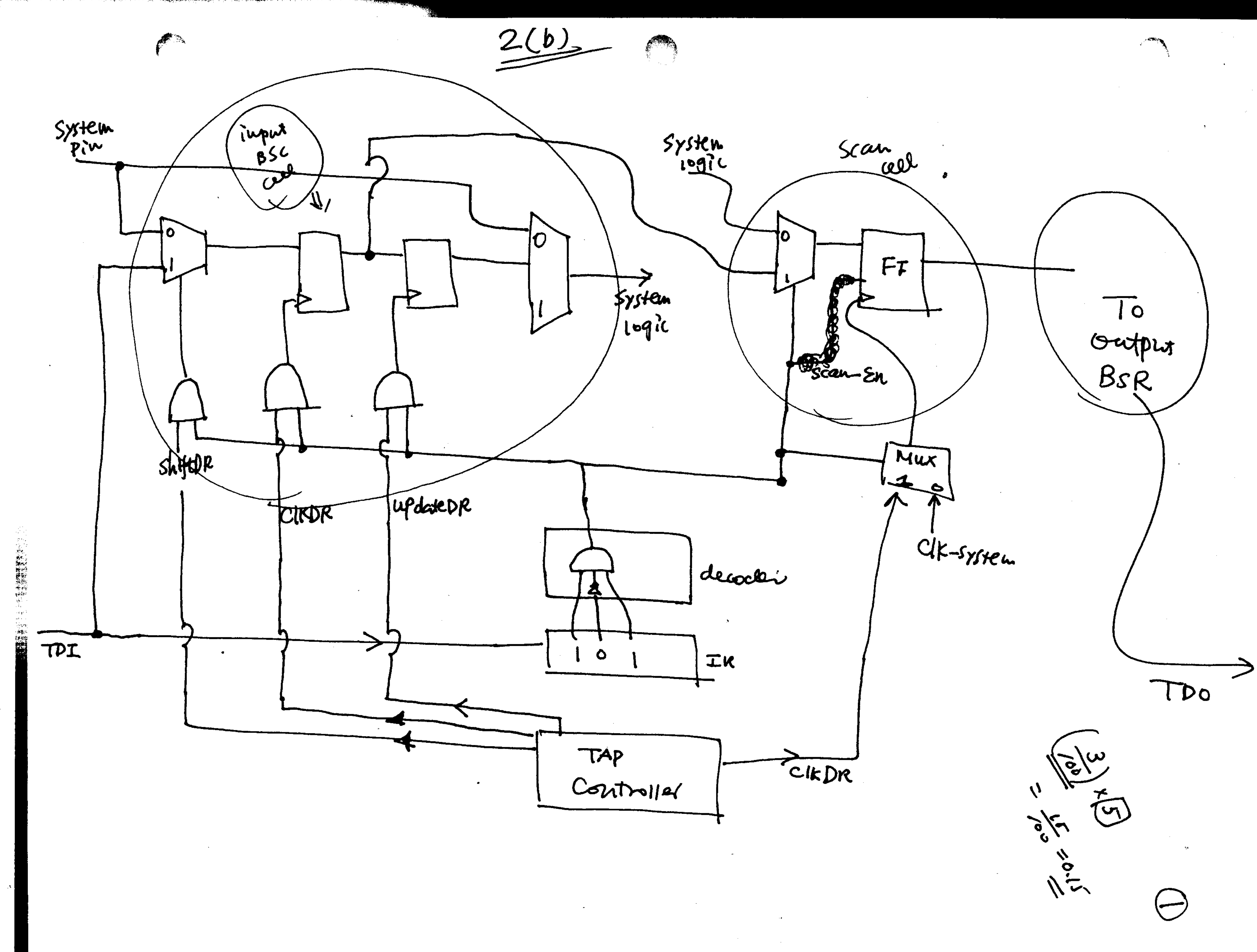


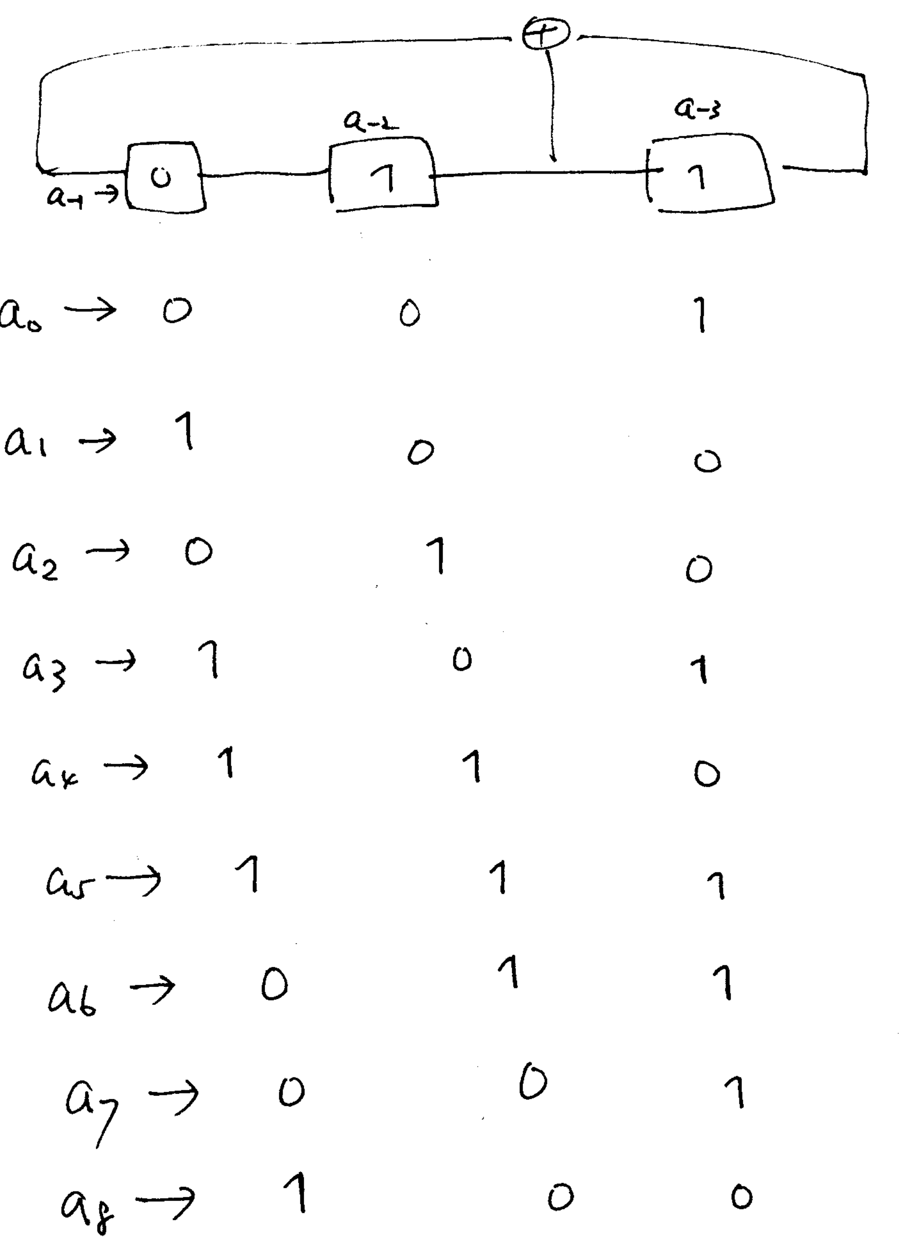
•



- Internal scan testing
- Instruction = 101

how to design the boundary scan ??


2(b)
System pin
input BSC cell
0
1
0
1
System logic
ShiftDR
ClkDR
UpdateDR
System logic
Scan cell
0
1
FF
Scan-En
To Output BSR
Mux
1
0
Clk-system
decoder
1 0 1
IR
TDI
TDO
TAP Controller
ClkDR
(3/100) x 5
= 15/100 = 0.15
1



a-2
a-3
a-1 → 0
1
1
a0 → 0 0 1
a1 → 1 0 0
a2 → 0 1 0
a3 → 1 0 1
a4 → 1 1 0
a5 → 1 1 1
a6 → 0 1 1
a7 → 0 0 1
a8 → 1 0 0

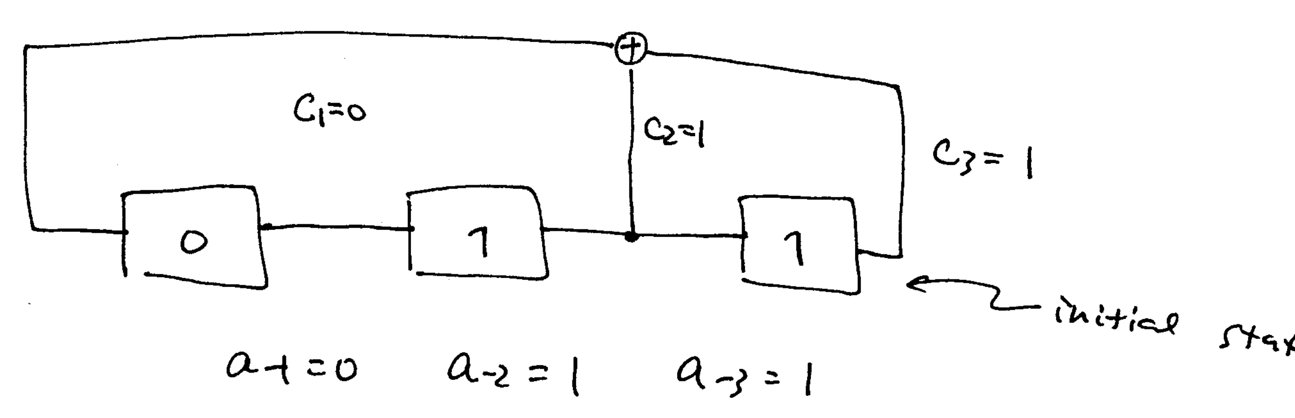
$$
\begin{array}{r|l}
 & x + x^3 + x^4 + x^5 + x^8 + \cdots \\
\hline
1+x^2+x^3 & x \\
 & x + x^3 + x^4 \\
\hline
 & x^3 + x^4 \\
 & x^3 + x^5 + x^6 \\
\hline
 & x^4 + x^5 + x^6 \\
 & x^4 + x^6 + x^7 \\
\hline
 & x^5 + x^7 \\
 & x^5 + x^7 + x^8 \\
\hline
 & x^8
\end{array}
$$

$a_0 x^0 + a_1 x^1 + \cdots$

$= (0, 1, 0, 1, 1, 1, 0, 0, 1, \cdots)$

characteristic polynomial → $p(x) = 1 + c_1 x^1 + c_2 x^2 + c_3 x^3$

$= 1 + x^2 + x^3$



$a_{-1} = 0 \quad a_{-2} = 1 \quad a_{-3} = 1$

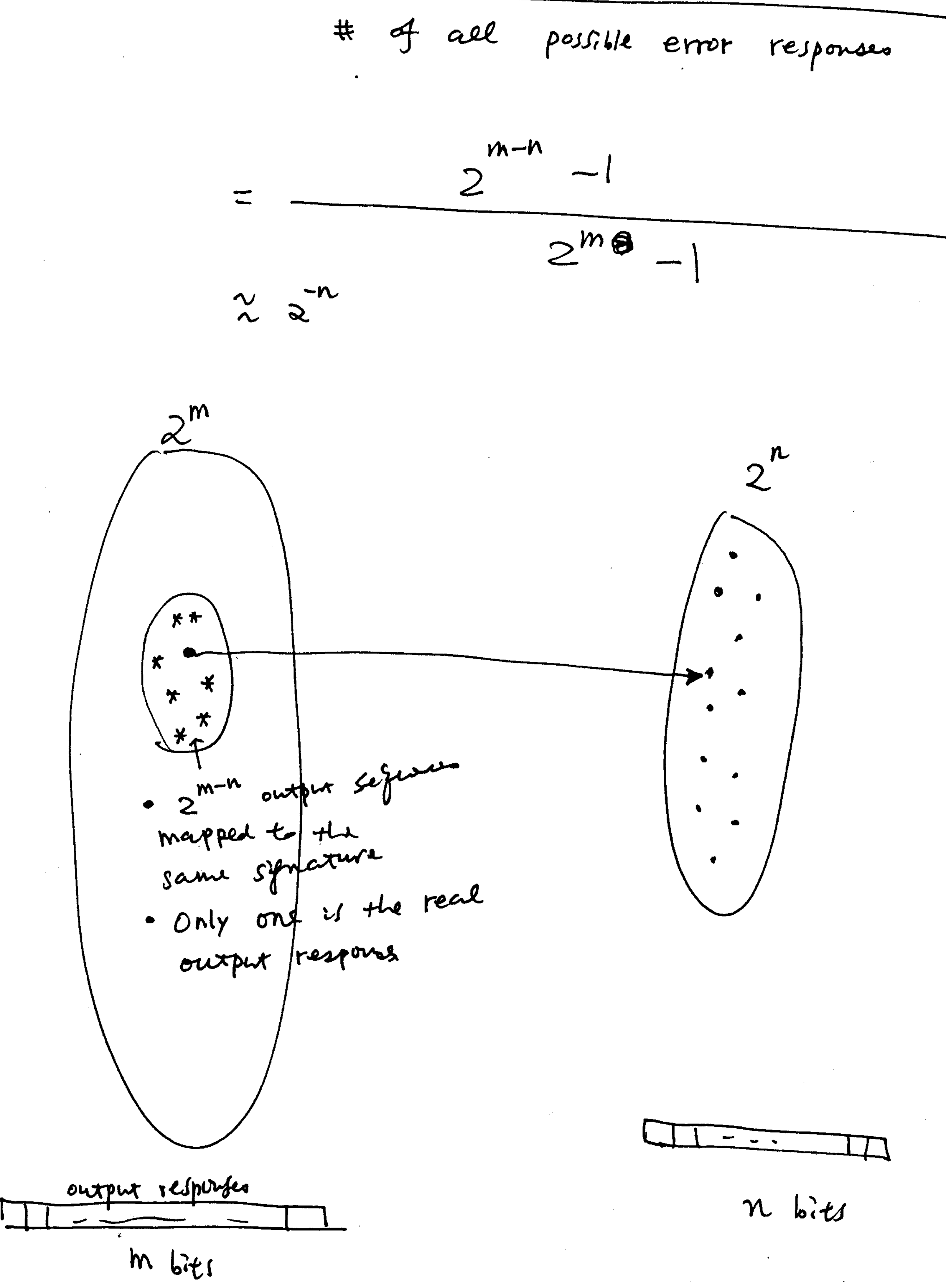
Initial polynomial → $I(x) = c_1 x^1 (a_{-1} x^{-1}) + c_2 x^2 (a_{-2} x^{-2} + a_{-1} x^{-1}) + c_3 x^3 (a_{-3} x^{-3} + a_{-2} x^{-2} + a_{-1} x^{-1})$

$= 0 + x^0 + x^0 + x^1 = x$

$$P(\text{aliasing}) = \frac{\#\text{ of error responses escape detection after compression}}{\#\text{ of all possible error responses}}$$

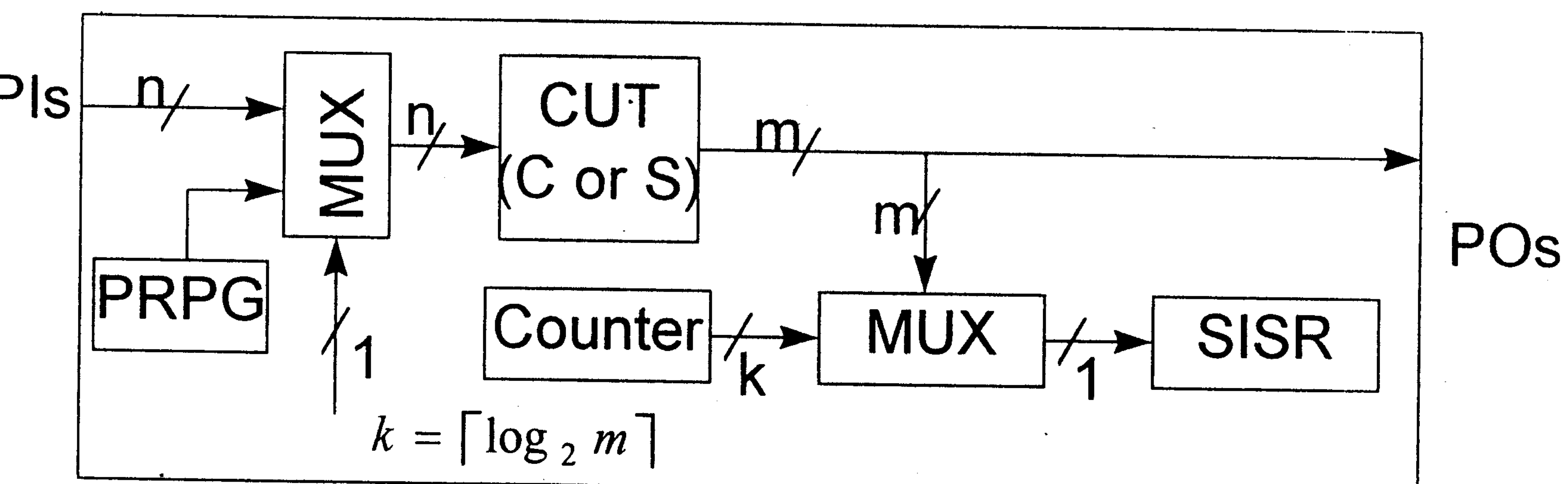
$$= \frac{2^{m-n} - 1}{2^{m} - 1}$$

$$\approx 2^{-n}$$



BIST - CSBL

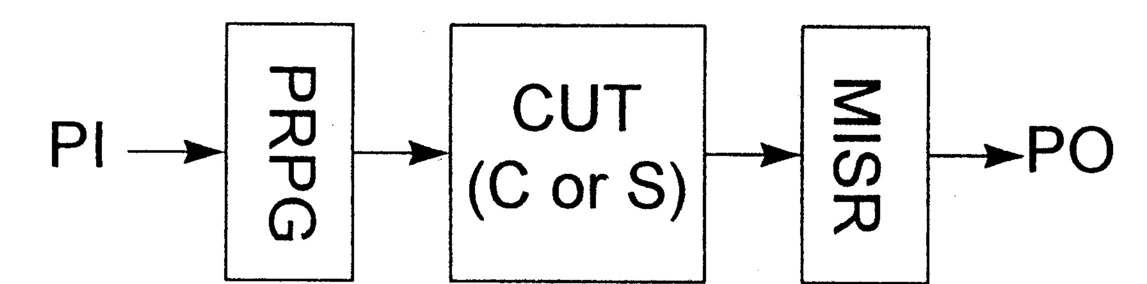
- Centralized and Separate Board-Level BIST [Benowitz 75]



- Use only one Signature Register
- Tests repeat m times to reduce hardware cost

BIST - BEST

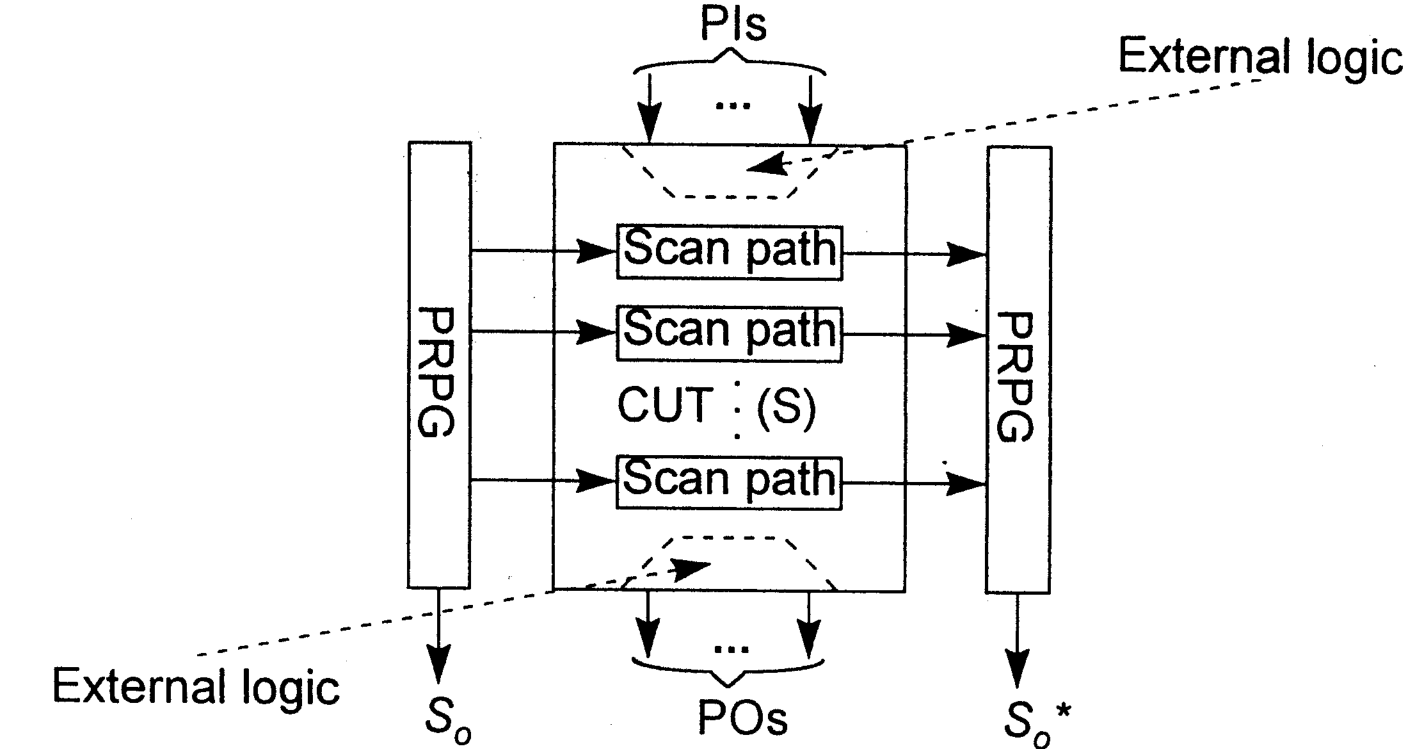
- Built - Evalution and Self Test [Resnick 83]



- Pseudo random testing
- Hardware overhead is low
- Test length can be long for CUT with random-pattern resistant faults.

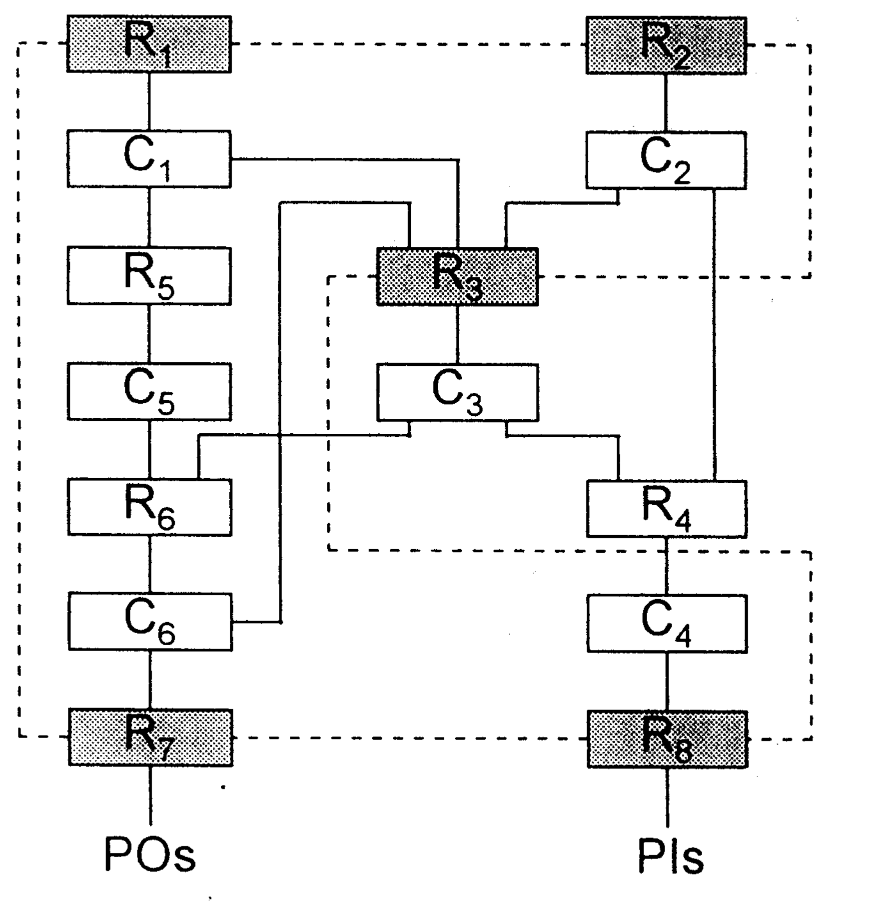
BIST - STUMPS

- Self-Test using MISR and Parallel SRSG [Bardell 82,84]

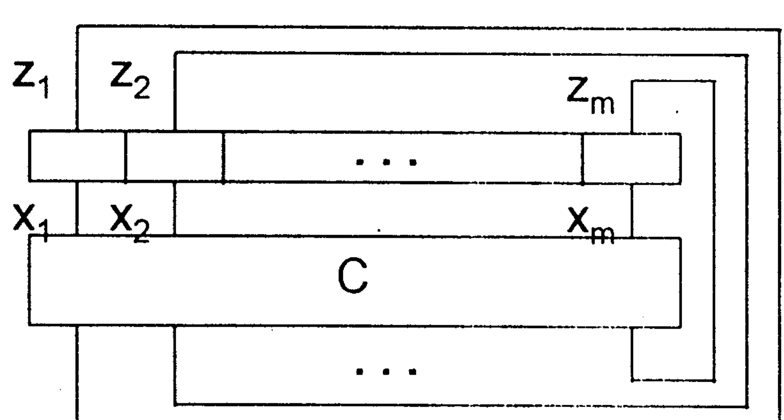


- Multiple scan chain to reduce test time

BIST - CSTP



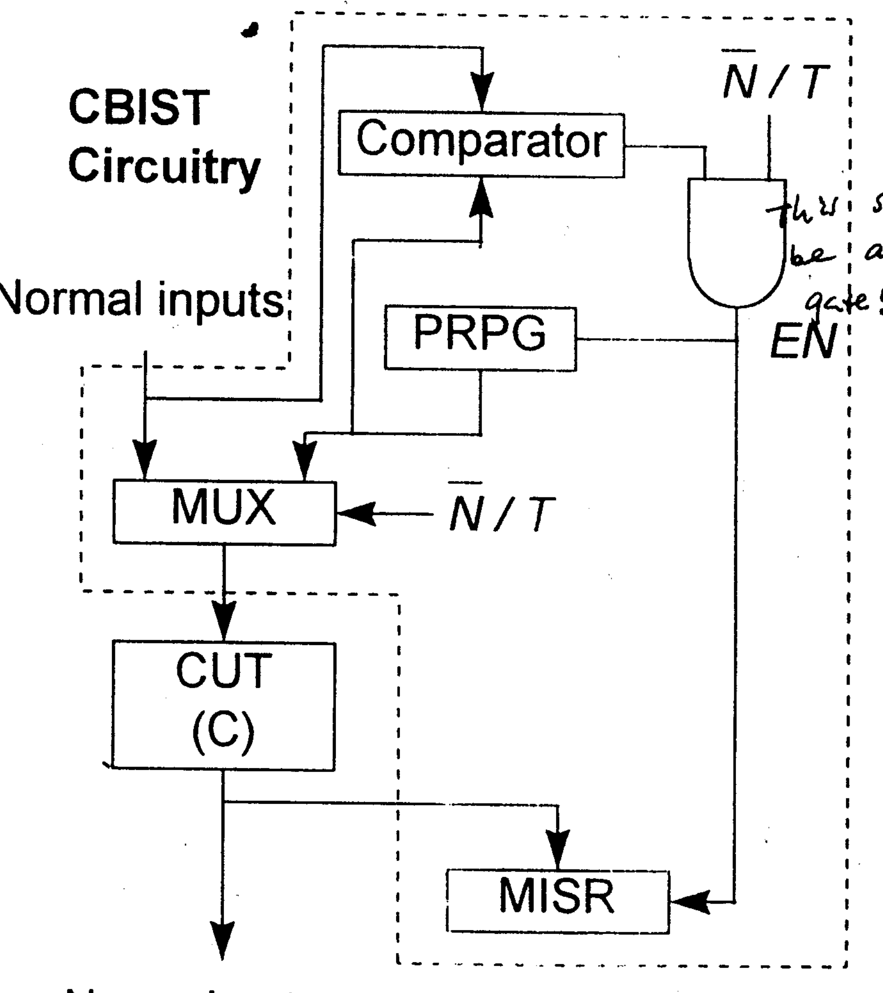
BIST - Example



- Circular Self Test Path [Krasniewski 89]
- Similar to SST except that boundaries are scanned
- Not all registers are self-test path registors.

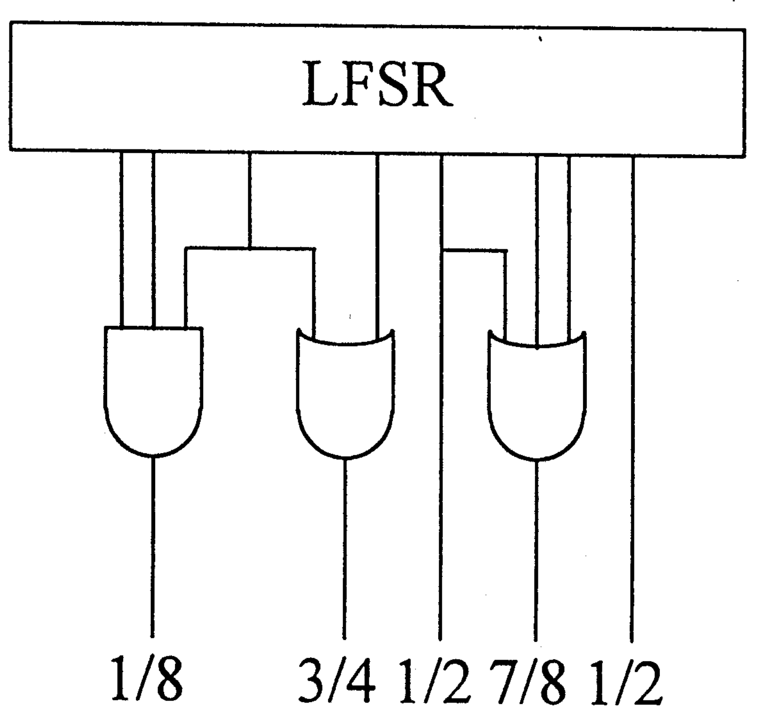
BIST - CBIST

- Concurrent Built-in Self Test [Saluja 88]
- Detect test patterns from normal inputs sequence
- Once a pattern is detected, compress the response and tick the test clock.
- If waited too long, insert a test pattern from PRPG.



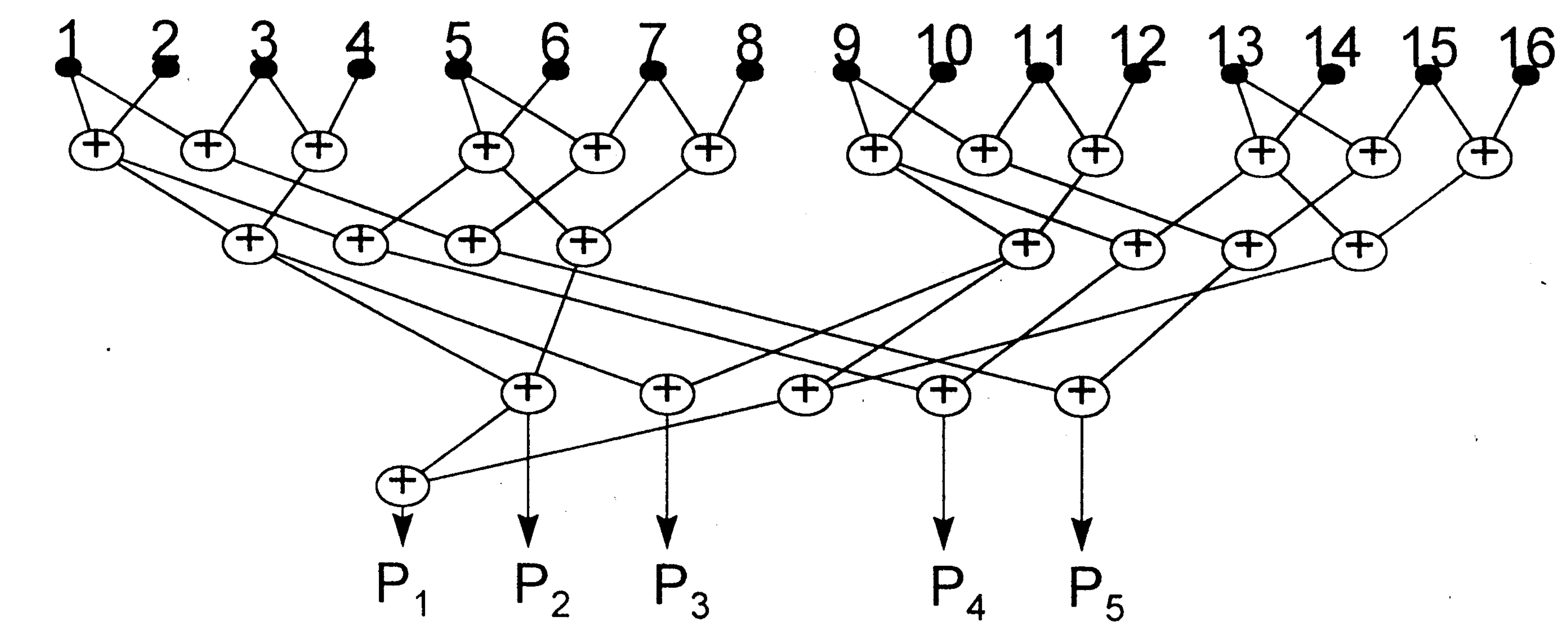
BIST - Weighted Pseudo Random Test

LFSR Based

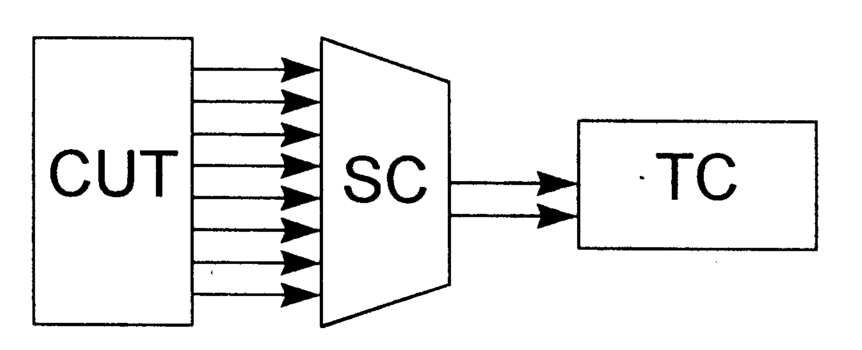


Space Compression

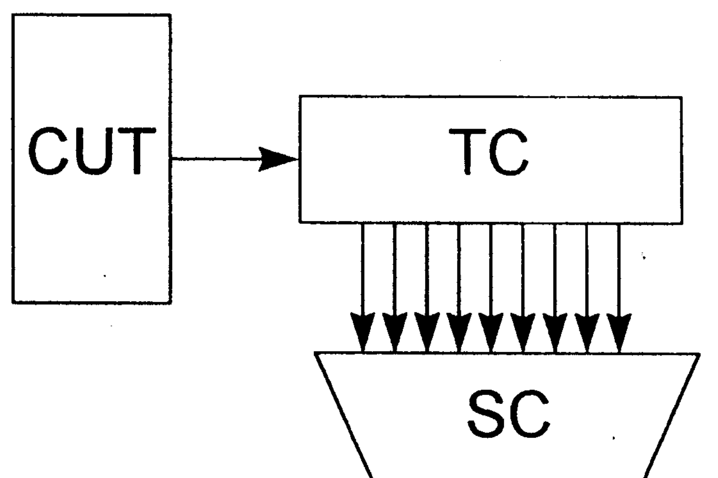
- Use space compression to handle large output circuits.
- Use XOR gates to compress space.
- Use Error Control Coding to achieve better fault coverage.
- Example: A 16 SEC-DED code compresses 16 outputs into 5.



BIST - Space and Time Compression



Space-Time Compression



Time-Space Compression

Functional Faults

* Cell stuck
* Read/write line stuck
* Data line stuck
* Driver stuck
* Chip-select line stuck
* Open in data line

---

* Short between data lines
* Crosstalk between data lines

---

* Address line stuck
* Open decoder
* Wrong access
* Open in address line
* Shorts between address lines
* Multiple access

---

* Cell can be set to 0 but not to 1 (or vice-versa)

---

* Pattern sensitive interaction between cells

Neighborhood Pattern Sensitive Fault

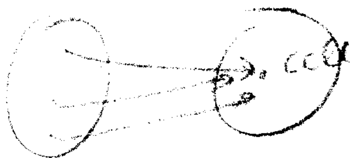
* A Pattern Sensitive Fault (PSF) is defined as follows: The contents of a cell, or the ability to change the contents, is influenced by the contents of all other cells in the memory.
* The PSF can be considered the most general case of the k-coupling fault.
* The PSF model allows the neighborhood to take on any position in the memory array.
* When the neighborhood is allowed to take on only a single position, one speaks about a Neighborhood Pattern Sensitive Fault (NPSF).

M

Address Decoder Fault (AF)

* Address decoder Faults (AFs) concern faults in the address decoder.
* Functional faults within the address decoder will result:

 1. With a certain address, no cell will be accessed.
 2. There is no address with which this cell can be accessed. A certain cell is never accessed.
 3. With a certain address, multiple cells are accessed simultaneously.
 4. A certain cell can be accessed with multiple addresses.



Relationship between Faults

Fault	Cause
SAF	* Cell stuck
SAF	* Driver stuck
SAF	* Read/write line stuck
SAF	* Chip-select line stuck
SAF	* Data line stuck
SAF	* Open in data line
CF	* Short between data lines
CF	* Crosstalk between data lines
AF*	* Address line stuck
AF	* Open in address line
AF	* Open decoder
AF	* Shorts between address lines
AF	* Wrong access
AF	* Multiple access
TF	* Cell can be set to 0 but not to 1 (or vice-versa)
NPSF	* Pattern sensitive interaction between cells

*AF: address decoder fault

M

March X

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

Faults detected:

AF, SAF, TF, CFins → 2 cases:

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

proof:

① SAF:

w0, r0
w1, r1 will detect

② TF:

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

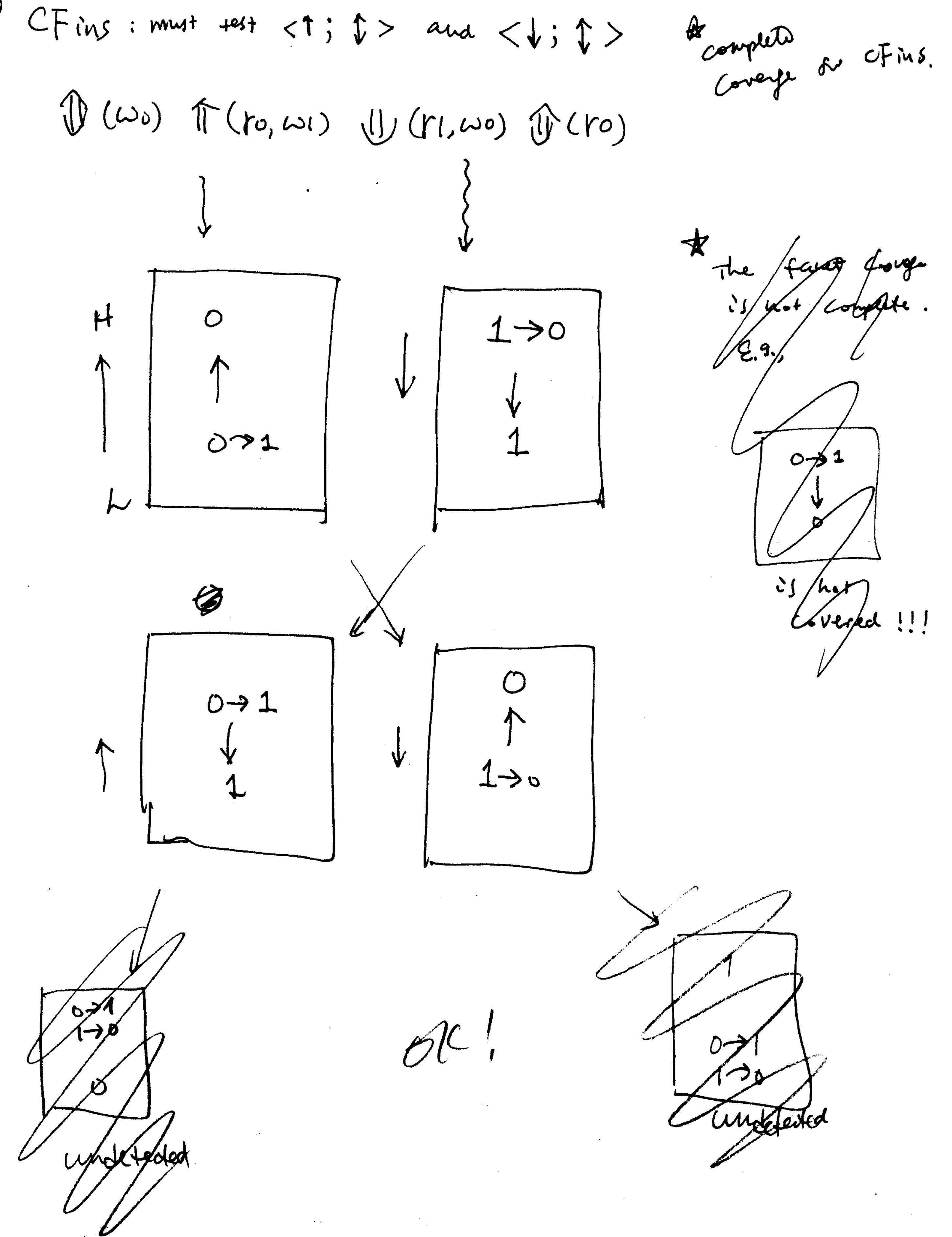
0→1 transition — check

1→0 transition — check.

③ CFins : must test $<\uparrow ; \updownarrow>$ and $<\downarrow ; \updownarrow>$

✪ complete Coverage of CFins.

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



④ AFs

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

* cell C is wrongly accessed if we access A

will be detected in

$\Uparrow(r0, w1)$

∴ when march to C, its value has been chaged to 1

But, r0 is expected.

$\Uparrow(r0, w1)$ will defect. when A read, it reads a "1". But, 0 is expected.

* If a cell is never accessed

If C is never accessed, then the value of c-1 will be given for the read ⇒ the error can be immediately detected.

Tests for Stuck-At, Transition and Coupling Faults

March alg.	Test len.	Fault coverage
MATS	4n	Some AFs, SAFs
MATS+	5n	AFs, SAFs
Marching 1/0	14n	AFs, SAFs, TFs
MATS++	6n	AFs, SAFs, TFs
March X	6n	AFs, SAFs, TFs, CFins
March C-	10n	AFs, SAFs, TFs, CFins, CFids
March A	15n	AFs, SAFs, TFs, CFins, L-CFids
March Y	8n	AFs, SAFs, CFins, TFs_l/w_CFins
March B	17n	AFs, SAFs, CFins, L_CFids, TFs_l/w_CFids

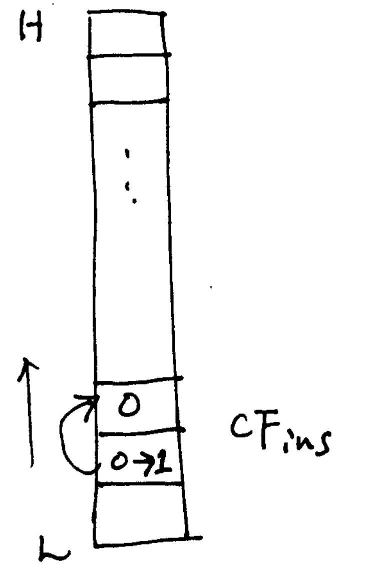
NPSF Test Patterns

Fault type	# of test patterns
ANPSF	(k-1)*2^k
PNPSF	2^k
APNPSF	k*2^k
SNPSF	2^k

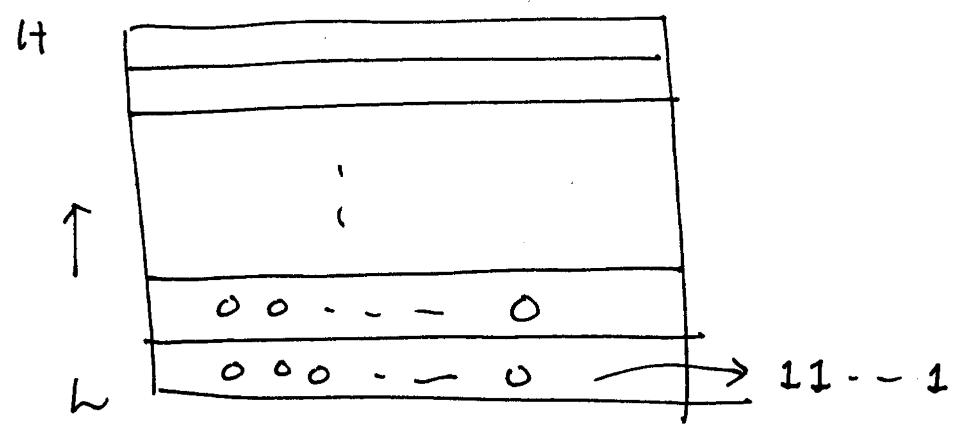
* k: The size of the neighborhood.
* ANPSF: Active Neighborhood Pattern Sensitive Fault
* PNPSF: Passive Neighborhood Pattern Sensitive Fault
* APNPSF: Active+Passive Neighborhood Pattern Sensitive Fault
* SNPSF: Static Neighborhood Pattern Sensitive Fault

Background Data

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



- Assume single bit in each memory word.
- However, what can we do if memory is accessed by word ?



Can we just replace 0 by 00---0 and

1 by 11--1 ?

- Assume $B=4$ (4 bits/word)

4

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

↓

$\Updownarrow(w0101);\ \Uparrow(r0101, w1010);\ \Downarrow(r1010, w0101);$

$\Updownarrow(r0101);$

0 ⇝ 0101

1 ⇝ 1010

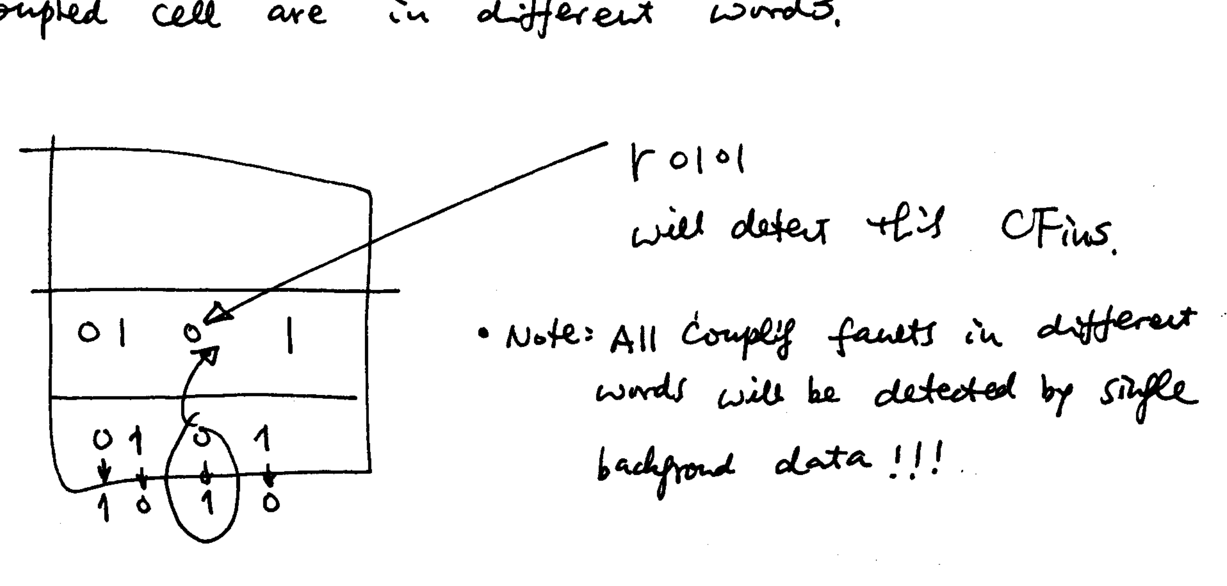
0101 is called <u>background data</u>

1010 is called <u>inverted background data</u>

- It can be proved that all stuck-at & transition faults can be detected.

- How about CF_{ins} ?

Yes, this fault can also be detected, if the coupling cell and coupled cell are in different words.



r0101
will detect this CF_{ins}.

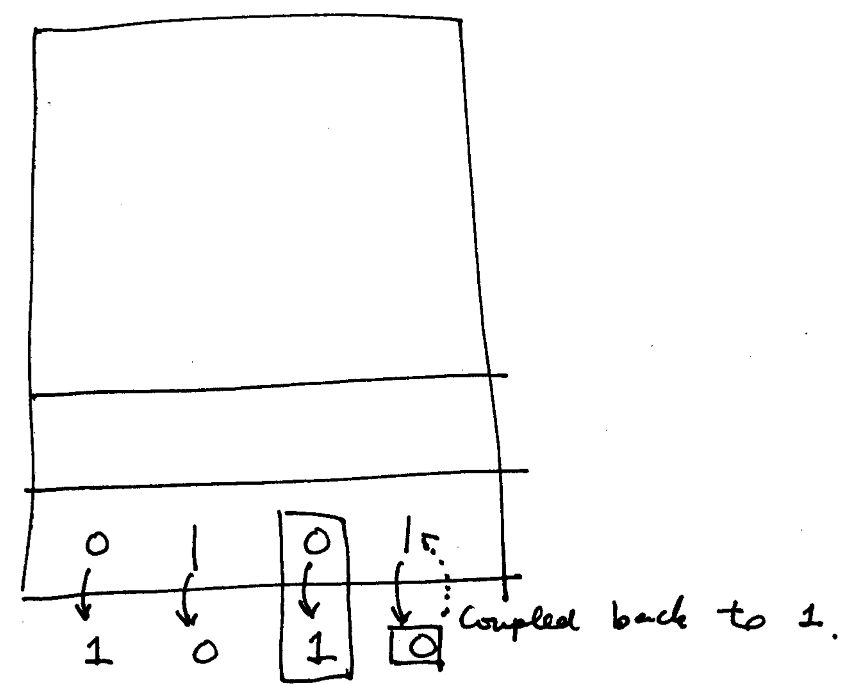
- Note: All coupling faults in different words will be detected by single background data !!!

- If the Coupling and Coupled cells are in the same word:

Case I: Write dominates the Coupling effect

The value specified in the write operation will be Stored, so CF will have no effect.

Case II: CF dominates the write operation



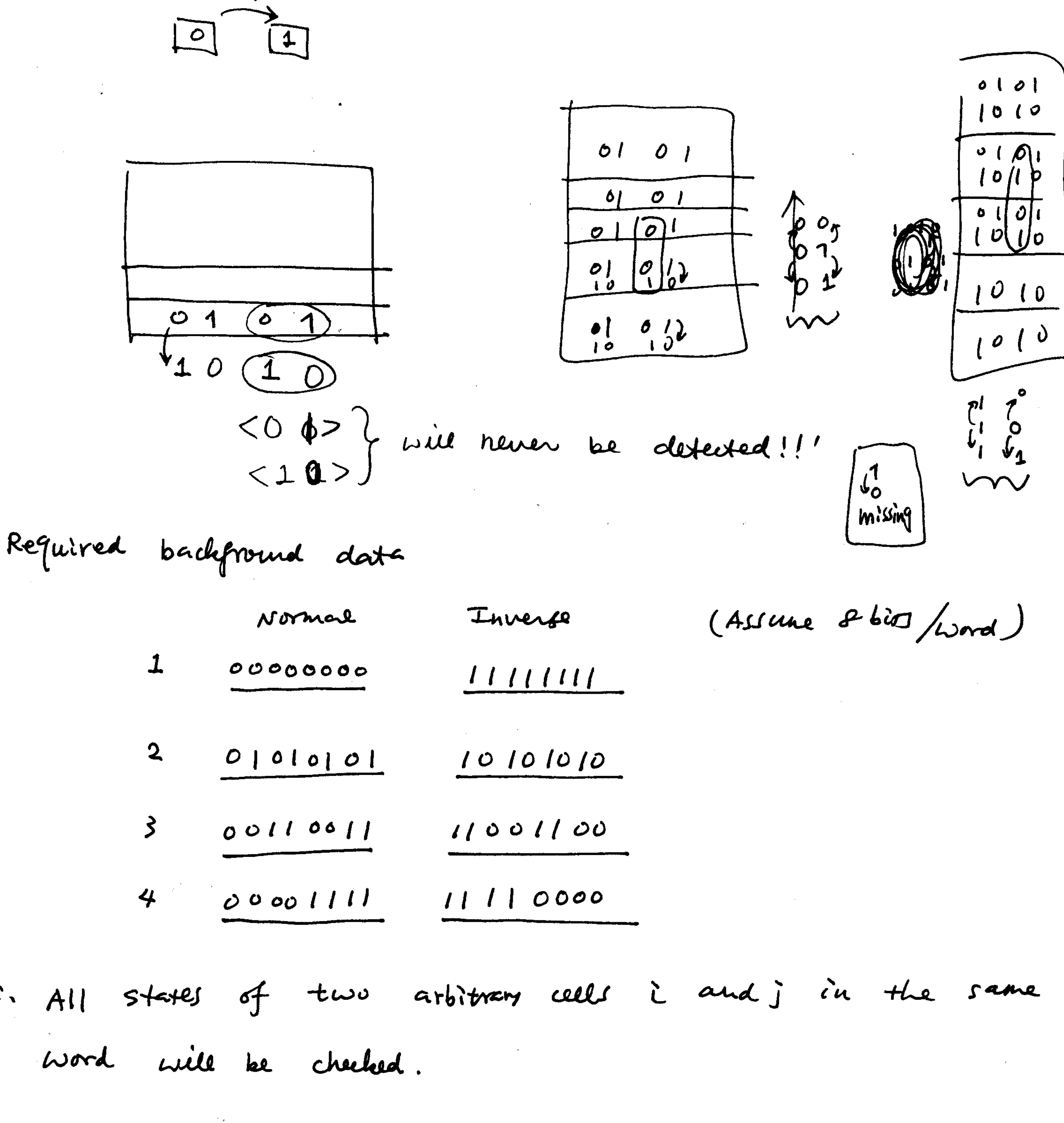
- This will be detected by the next read operation followed.

Lucky Case!!!

∴ A single backgrond data detects all CF_{ins}, SA, TF faults for word-oriented memory.

State Coupling fault (SCF)

- $<0;0>$, $<0;1>$, $<1;0>$, $<1;1>$



$<0\ \phi>$, $<1\ 0>$ } will never be detected!!!

- Required background data

	Normal	Inverse
1	00000000	11111111
2	01010101	10101010
3	00110011	11001100
4	00001111	11110000

(Assume 8 bits/word)

∴ All states of two arbitrary cells i and j in the same word will be checked.