Content Addressable Memory Testing

Jin-Fu Li
Department of Electrical Engineering
National Central University
Jungli, Taiwan

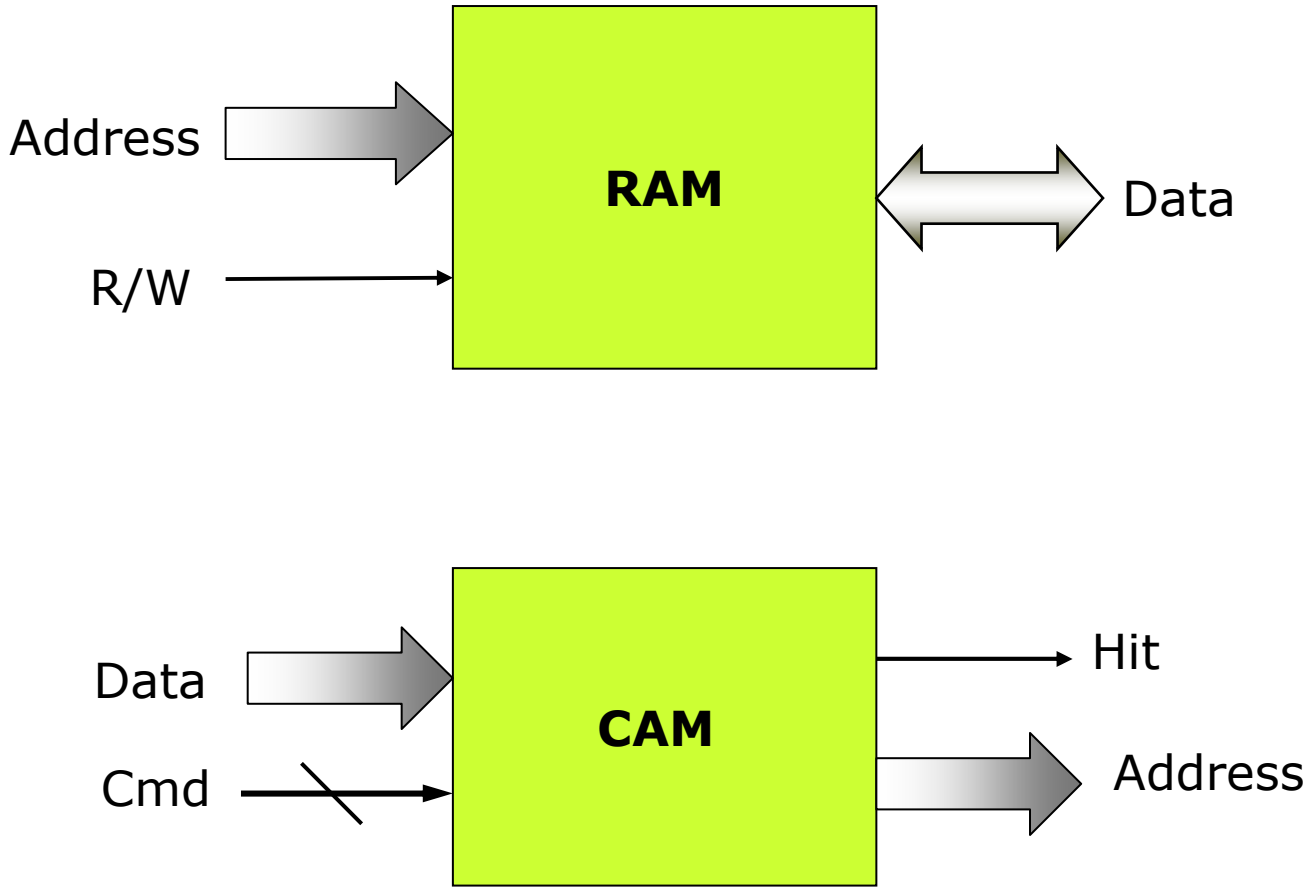
Outline

- Introduction
- Content Addressable Memories
- Comparison Faults of Binary CAMs
- Test Algorithms for Binary CAMs
- Testing Priority Encoder Faults of Binary CAMs
- Conclusions

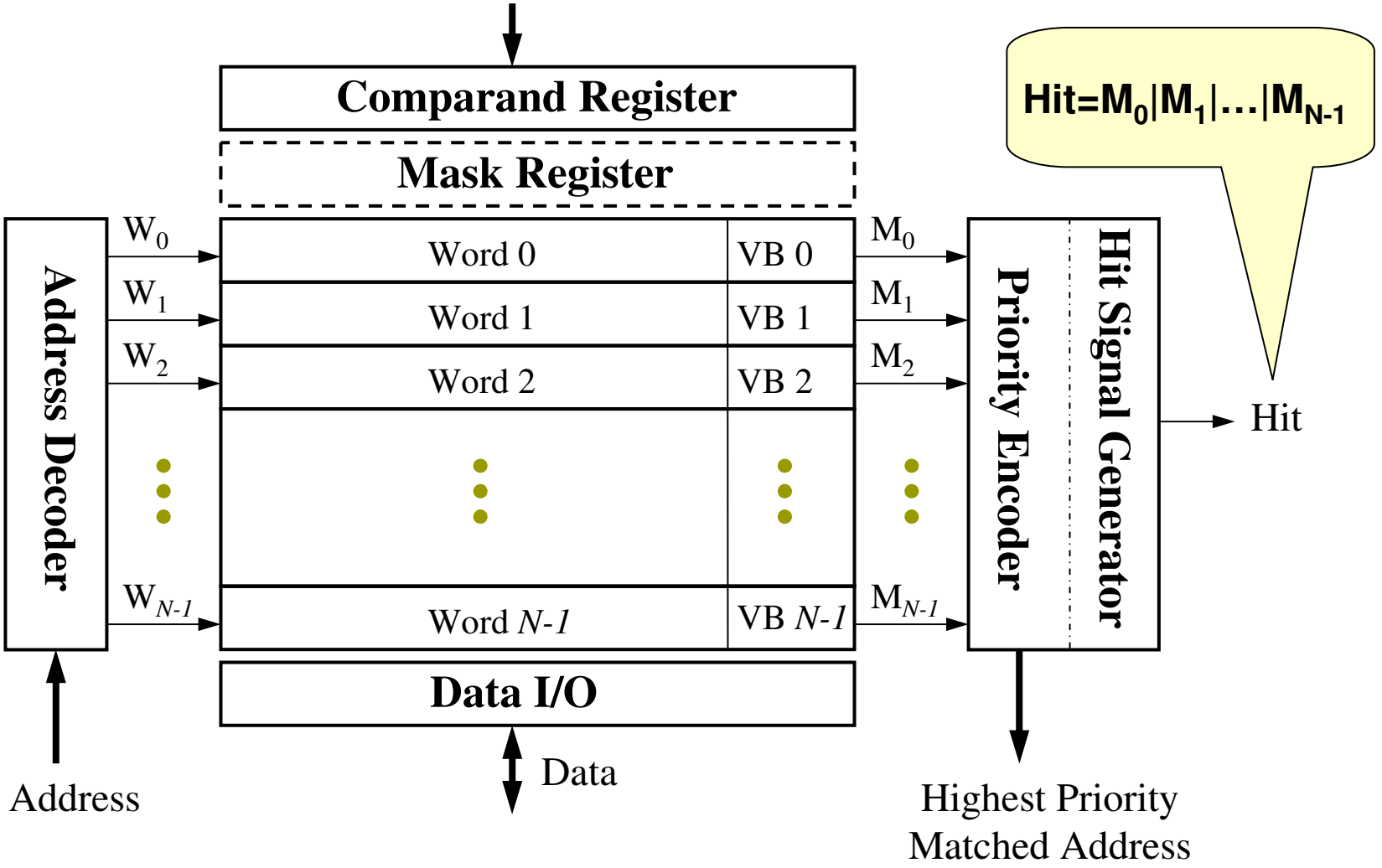
Introduction

- Content addressable memories (CAMs) are widely used in digital systems
  - E.g., router, cryptography, etc.
- With the advent technology, large CAMs can be embedded into system-on-chips (SOCs)
  - Performance and power consumption of CAMs can significantly be improved
- However, testing embedded CAMs is difficult due to poor accessability
- Moreover, CAM testing is more difficult than RAM testing, since CAM cell structure is more complicated

Difference Between RAM and CAM



Typical CAM Architecture

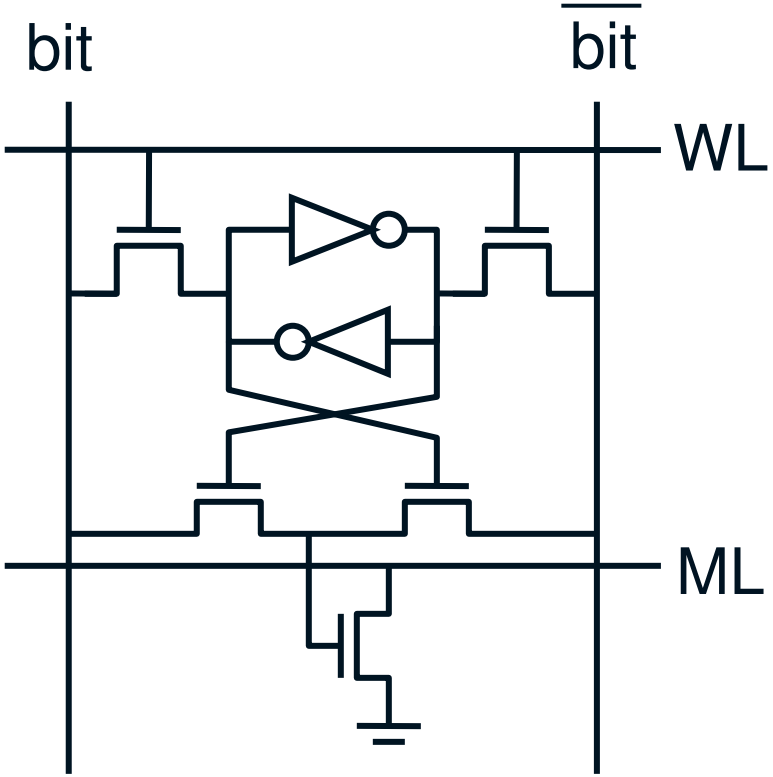


Basic Components

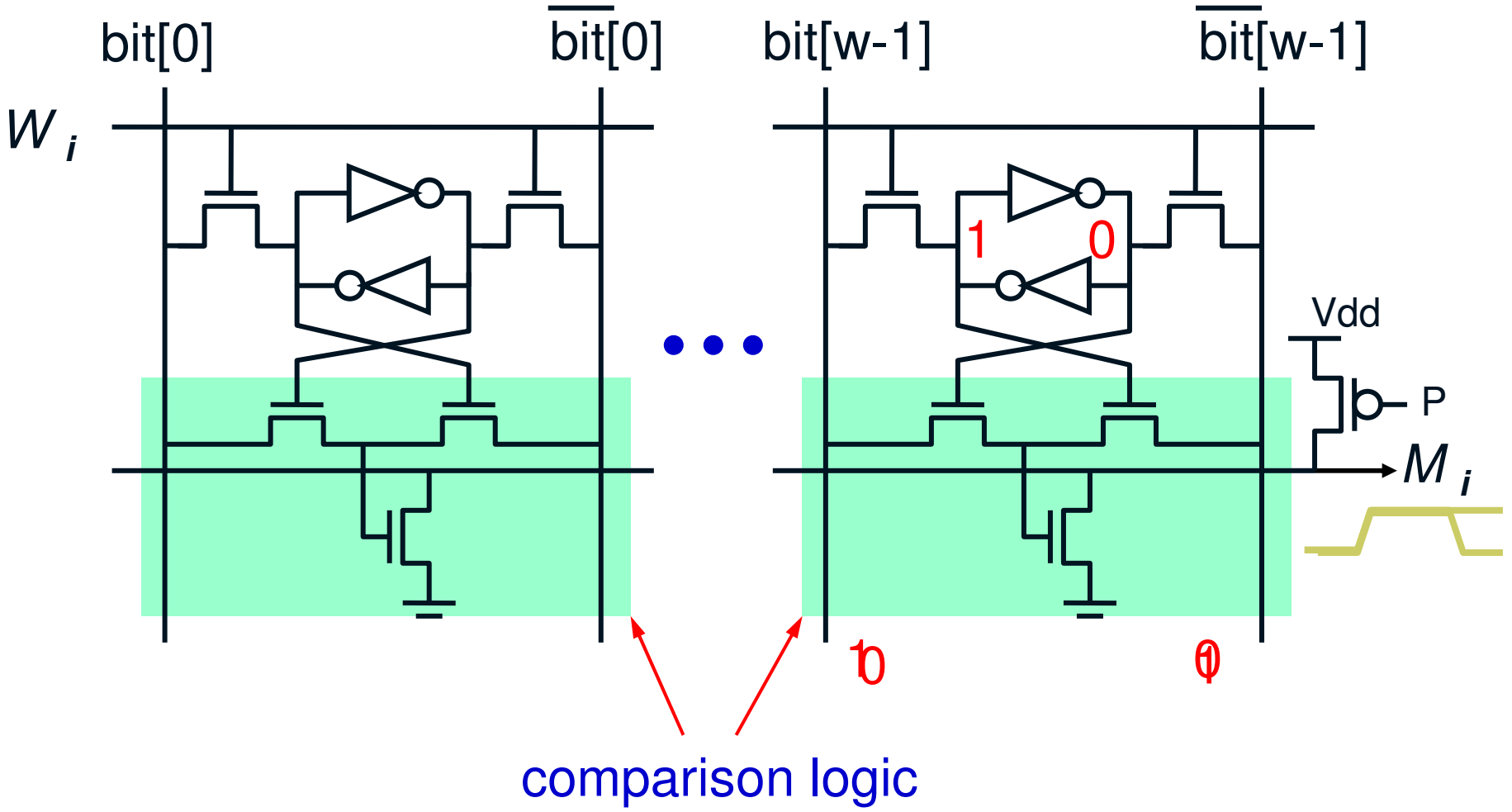
- Comparand Register
  - It contains the data to be compared with the content of the memory array
- Mask Register
  - It is used to mask off portions of the data word(s) which do not participate in the operations
- Memory Array
  - It provides storage and search (compare) medium for data
- Hit Signal Generator/Priority Encoder
  - It indicates success or failure of a compare operation

Binary CAM (BCAM)

- Binary CAM
  - It is comprised of binary CAM cells and each cell only can store “0” or “1”
- A typical NOR-type binary CAM cell



An NOR-Type BCAM Word



CAM Functional Faults

- Functional faults of CAM
  - RAM faults
  - Comparison faults
- Comparison fault
  - Fault effect is observed by the output of comparison results
- CAM basic operations
  - Write and Compare operations only
    - Testing is more difficult since the observability is poor
  - Write, Read, and Compare operations
    - Testing is more easy since the observability is good

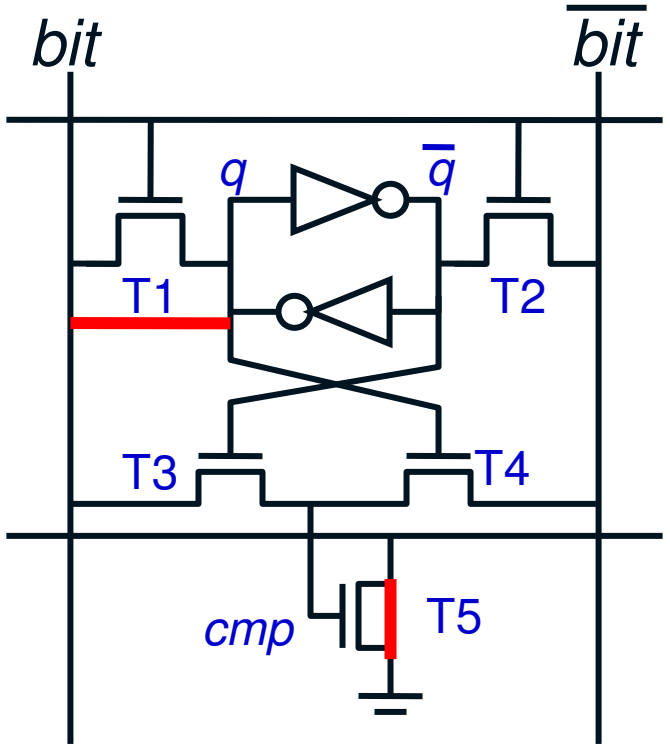
Comparison Faults of BCAMs

- Comparison faults of BCAM
  - Stuck-Match Fault (SMF)
  - Stuck-Mismatch Fault (SMMF)
  - Conditional-Match Fault (CMF)
  - Partial-Match Fault (PMF)
  - Equivalence-Mismatch Fault (EMMF)
  - Inequivalence-Match Fault (IMF)

[Source: K.-J. Lin and C.-W. Wu, IEEE TCAD00]

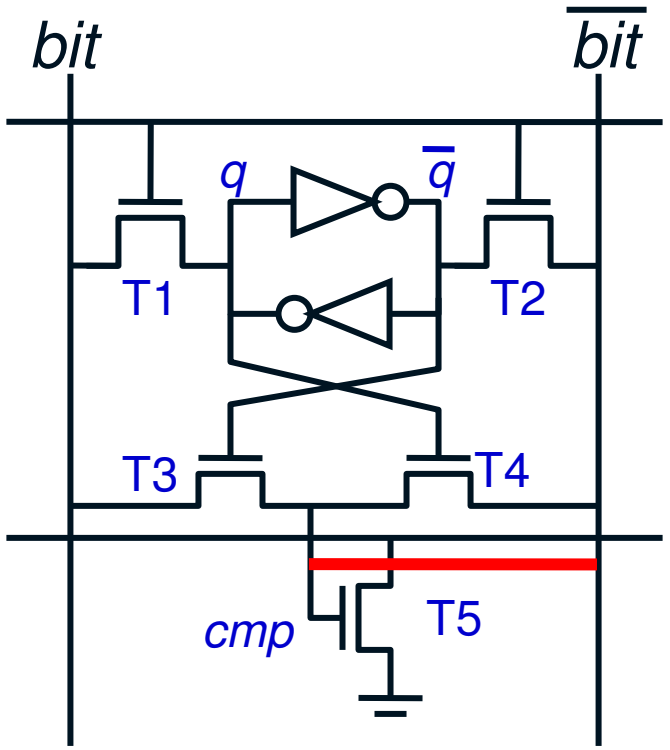
SMF & SMMF

- SMF (SMMF): a cell always matches (mismatches) the corresponding input bit irrespective of the CAM cell state and input pattern
- E.g., *q* and *bit* short
  - SMF
- E.g., T5 stuck-on
  - SMMF



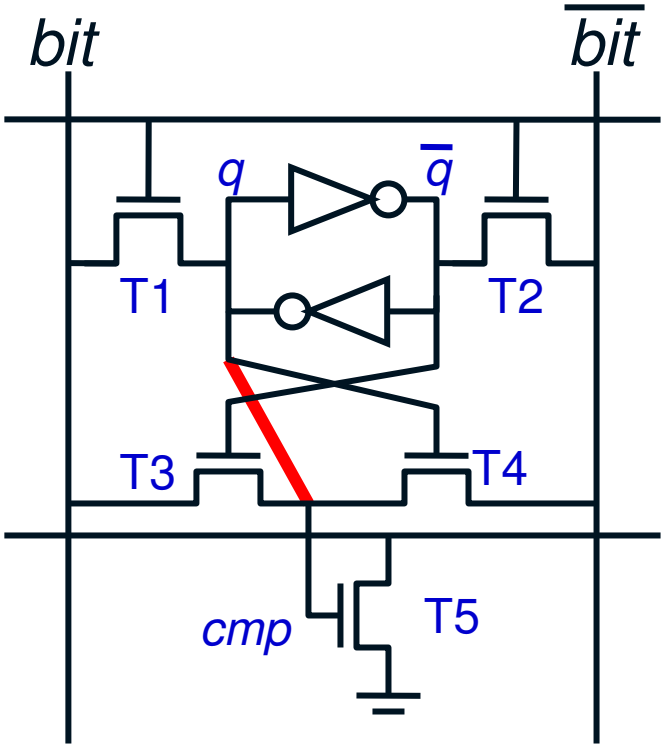
CMF

- A cell function is correct if it stores a logic value *x* (either 0 or 1), but it always provides an incorrect result for the subsequently Compare operations if it stores *x'*
- E.g., *bit'-cmp* short
  - CM1F



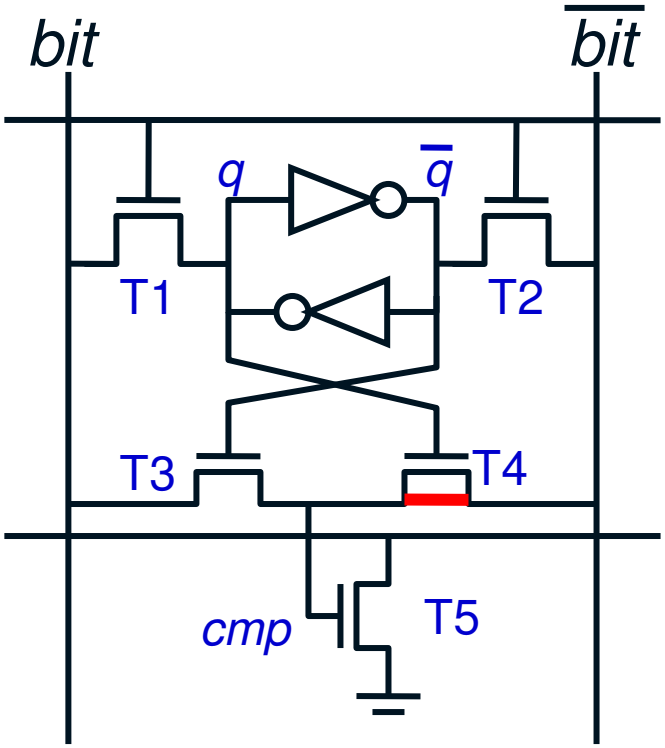
PMF

- A cell is stuck-matched for all subsequent Compare operations when a logic value *x* is written into the cell, and stuck-mismatched when *x'* is written into it
- E.g., *q* and *cmp* short
  - PM0F



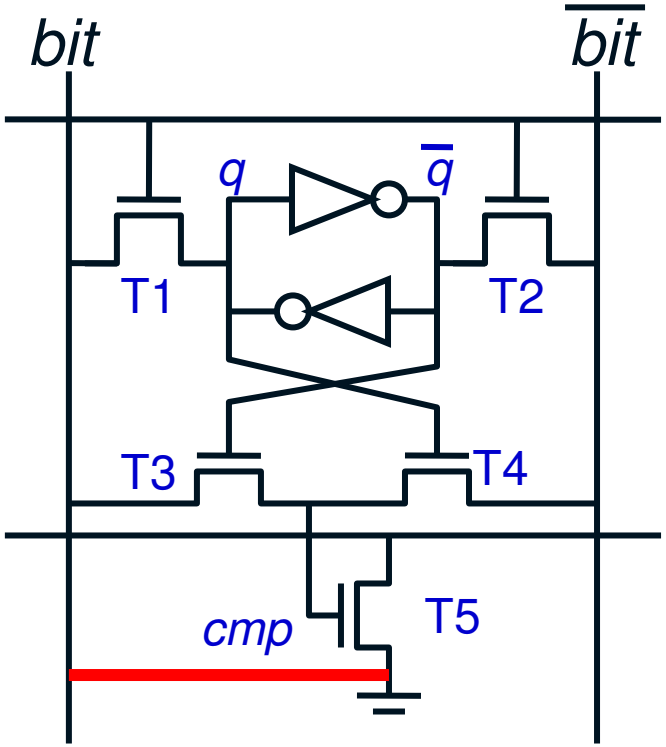
EMMF

- The Compare operation fails if the CAM cell stores a value *x* and is compared with the same input value *x*
- E.g., T4 stuck-on
  - EMM0F



IMF

- The Compare operation fails if the CAM cell stores a value *x* and is compared with the complementary input value *x'*
- E.g., *bit* and GND short
  - IM0F



Cell Response of Comparison Faults

- Cell responses of a BCAM cell executing compare-after-write operation

	w0, c0	*w0, c1*	*w1, c0*	*w1, c1*
Fault Free	M	MM	MM	M
SMF	M	M	M	M
SMMF	MM	MM	MM	MM
CM1F	MM	M	MM	M
CM0F	M	MM	M	MM
PM1F	MM	MM	M	M
PM0F	M	M	MM	MM
EMM1F	M	MM	MM	MM
EMM0F	MM	MM	MM	M
IM1F	M	MM	M	MM
IM0F	M	M	MM	M

[Source: K.-J. Lin and C.-W. Wu, IEEE TCAD00]

Notation for Test Algorithms

- *wA*: write an input pattern *A* to a given word and validate the word
- *rA*: read an expect data A from the addressed word
- cP_A: compare an input pattern P_A with all words
- *E* (Erase): reset the valid bit of a specified word, i.e., invalidate the word
- A masked input pattern is an input pattern filtered by a mask pattern
  - Every masked input pattern has the form P_A^M, where P^M is the mask pattern

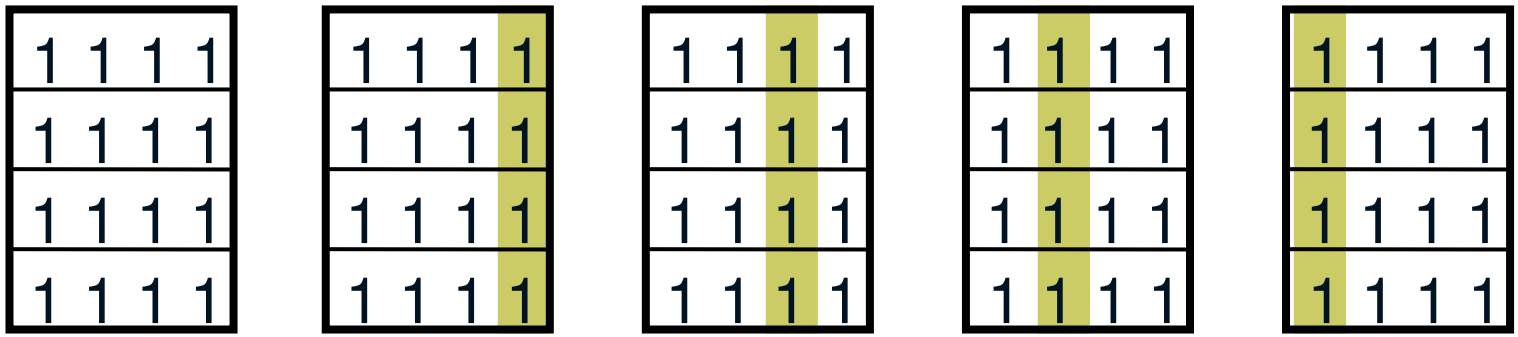
March-Like Test Algorithms

- Test algorithms
  - $T^1_{CAM} = \{\Updownarrow (w1); (cP_0^{\omega(0)}, cP_0^{\omega(1)}, \ldots, cP_0^{\omega(W-1)})\}$
  - $T^2_{CAM} = \{\Updownarrow (w0, cP_0, w1)\}$
  - $T^3_{CAM} = \{\Updownarrow (w0); (cP_1^{\omega(0)}, cP_1^{\omega(1)}, \ldots, cP_1^{\omega(W-1)})\}$
  - $T^4_{CAM} = \{\Updownarrow (w1, cP_1, w0)\}$
  - where $\omega(i) = 2^W - 1 - 2^i$ for a CAM with *W*-bit words

[Source: K.-J. Lin and C.-W. Wu, IEEE TCAD00]

Faults Detected by Test 1

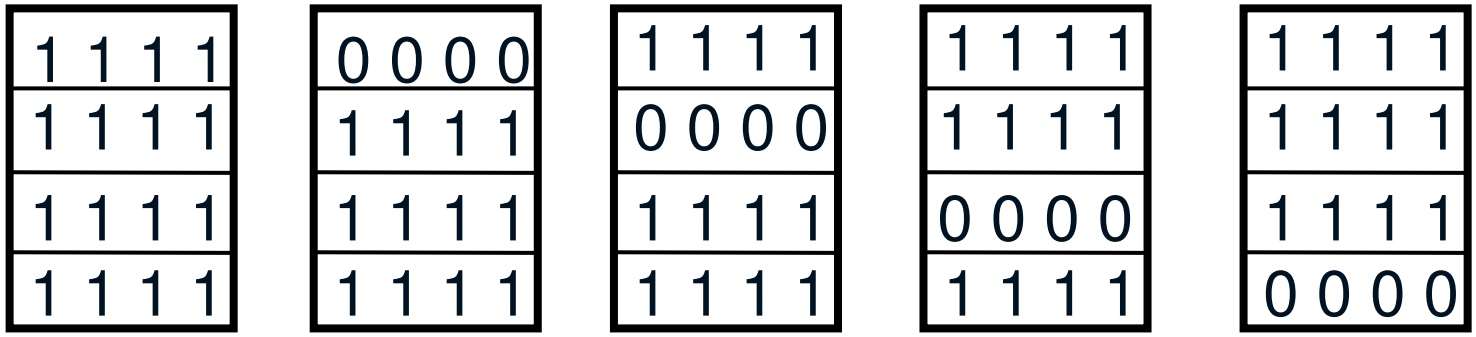
- $T^{1}_{CAM} = \{\Updownarrow(w1); (cP_0^{\omega(0)}, cP_0^{\omega(1)}, \ldots, cP_0^{\omega(W-1)})\}$



- Detected faults
  - SMF
  - CM0F
  - IM1F

Faults Detected by Test 2

- $T^{2}_{CAM} = \{\Updownarrow (w0, cP_0, w1)\}$



- Detected faults
  - SMMF
  - PMF
  - EMM0F

CAM Test with Read & Compare

- If a CAM has both Read and Compare operations, the test length can be reduced
  - $MLT-1 = \{\Updownarrow(w1); \Uparrow(w0, cP_0, w1); \Uparrow(r1, w0); (cP_1^{\omega(0)}, cP_1^{\omega(1)}, \ldots, cP_1^{\omega(W-1)}); \Downarrow(w1, cP_1, w0); \Downarrow(r0, w1); (cP_0^{\omega(0)}, cP_0^{\omega(1)}, \ldots, cP_0^{\omega(W-1)})\}$
- The intra-word coupling faults are not fully covered by the MLT-1
- The proposed MLT-2 test algorithm for the intra-word CFs is as shown below
  - $MLT-2 = \{\Updownarrow(wD); \Updownarrow(w\overline{D}); (cP_D^{\omega(0)}, cP_D^{\omega(1)}, \ldots, cP_D^{\omega(W-1)}); \Updownarrow(wD); (cP_{\overline{D}}^{\omega(0)}, cP_{\overline{D}}^{\omega(1)}, \ldots, cP_{\overline{D}}^{\omega(W-1)})\}$

    where D={0011,0101} for 4-bit words

[Source: J.-F. Li, R.-S. Tzeng and C.-W. Wu , JETTA03]

Faults Detected by MLT-1

Fault-free status when element $\Uparrow (w0, cP_0, w1)$ is executed

Operation		Content (M_i)		
		W_0 addressed	W_1 addressed	W_2 addressed
$w\mathbf{0}$	W_0	000 (x)	111 (x)	111 (x)
	W_1	111 (x)	000 (x)	111 (x)
	W_2	111 (x)	111 (x)	000 (x)
$cP_\mathbf{0}$	W_0	000 (1)	111 (0)	111 (0)
	W_1	111 (0)	000 (1)	111 (0)
	W_2	111 (0)	111 (0)	000 (1)
$w\mathbf{1}$	W_0	111 (x)	111 (x)	111 (x)
	W_1	111 (x)	111 (x)	111 (x)
	W_2	111 (x)	111 (x)	111 (x)

This test element can detect the SAFs and interword CFs whose victims are forced to 1. The aggressors of the CFs undergo the $\uparrow$ transition and are located at lower addresses than the victims. Also, the SMMF, PMF, EMMF, and XMMF can be detected since they cause the match signals to become mismatched and the Hit output is 0.

[Source: J.-F. Li, R.-S. Tzeng and C.-W. Wu , JETTA03]

Faults Detected by MLT-1

Fault-free status when element $\Uparrow (r1, w0)$ is executed

Operation		Content (M_i)		
		W_0 addressed	W_1 addressed	W_2 addressed
$r\mathbf{1}$	W_0	111 (x)	000 (x)	000 (x)
	W_1	111 (x)	111 (x)	000 (x)
	W_2	111 (x)	111 (x)	111 (x)
$w\mathbf{0}$	W_0	000 (x)	000 (x)	000 (x)
	W_1	111 (x)	000 (x)	000 (x)
	W_2	111 (x)	111 (x)	000 (x)

This test element can detect the inter-word CFid where the aggressors are located at higher addresses than the victims and the inter-word CFid where the aggressors are located at lower addresses than the victims. SAF(0) and inter-word CFst where the aggressors are in state 0 and at lower addresses than the victims or in state 1 and at higher addresses than the victims also can be detected.

[Source: J.-F. Li, R.-S. Tzeng and C.-W. Wu , JETTA03]

Faults Detected by MLT-1

Fault-free status when element $(cP_1^{\omega(0)}, cP_1^{\omega(1)}, \ldots, cP_1^{\omega(W-1)})$ is executed

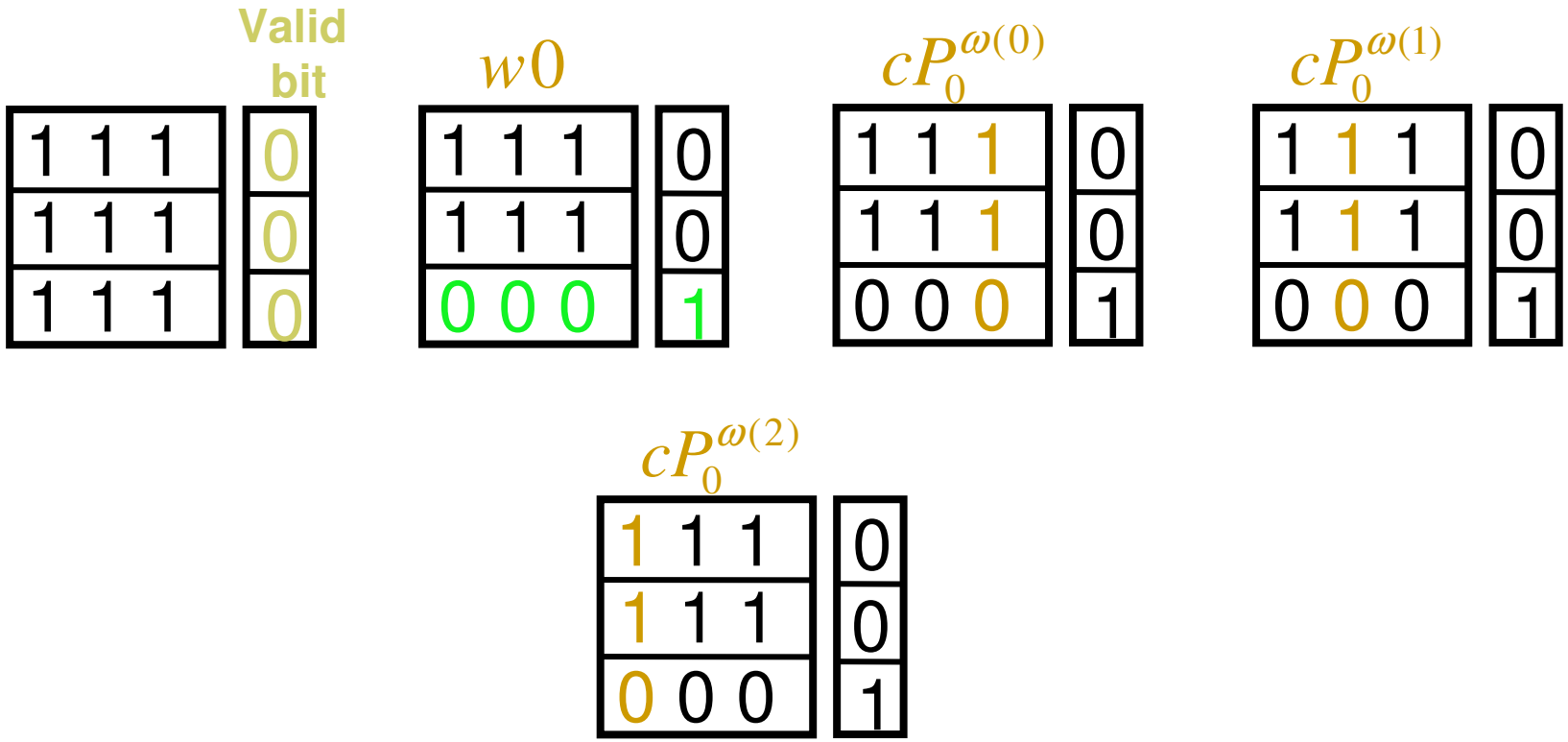
	Content (M_i)		
	cP_1^6	cP_1^5	cP_1^3
W_0	000 (0)	000 (0)	000 (0)
W_1	000 (0)	000 (0)	000 (0)
W_2	000 (0)	000 (0)	000 (0)

This test element can detect the SMF, CMF, and IMF. Any other fault that forces the cell state to become 1 can also be detected, e.g., SAF(1) and the CF whose aggressor is 0 or undergoes a $\downarrow$ transition to turn the victim into 1.

[Source: J.-F. Li, R.-S. Tzeng and C.-W. Wu , JETTA03]

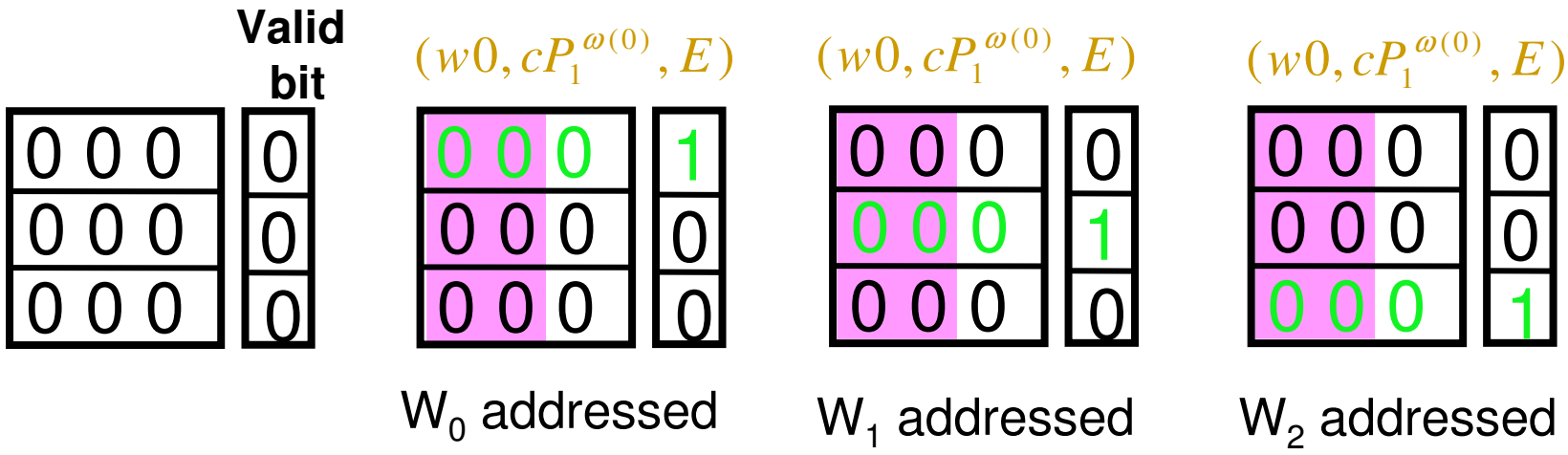
Fault-Location Tests

- Algorithms for locating the faulty cells in a row
  - $FLR-0=\{\Updownarrow(E);w0;(cP_0^{\omega(0)},cP_0^{\omega(1)},\ldots,cP_0^{\omega(W-1)})\}$
  - $FLR-1=\{\Updownarrow(E);w1;(cP_1^{\omega(0)},cP_1^{\omega(1)},\ldots,cP_1^{\omega(W-1)})\}$



Fault-Location Tests

- Algorithms for locating the faulty cells in a column
  - $FLC-0=\{\Updownarrow(E);(w0,cP_1^M,E)\}$
  - $FLC-1=\{\Updownarrow(E);(w1,cP_0^M,E)\}$
  - *M* is the same as the mask pattern of the Compare operation detecting the faulty column
- For example



Experimental Results

- Fault coverage for comparison faults

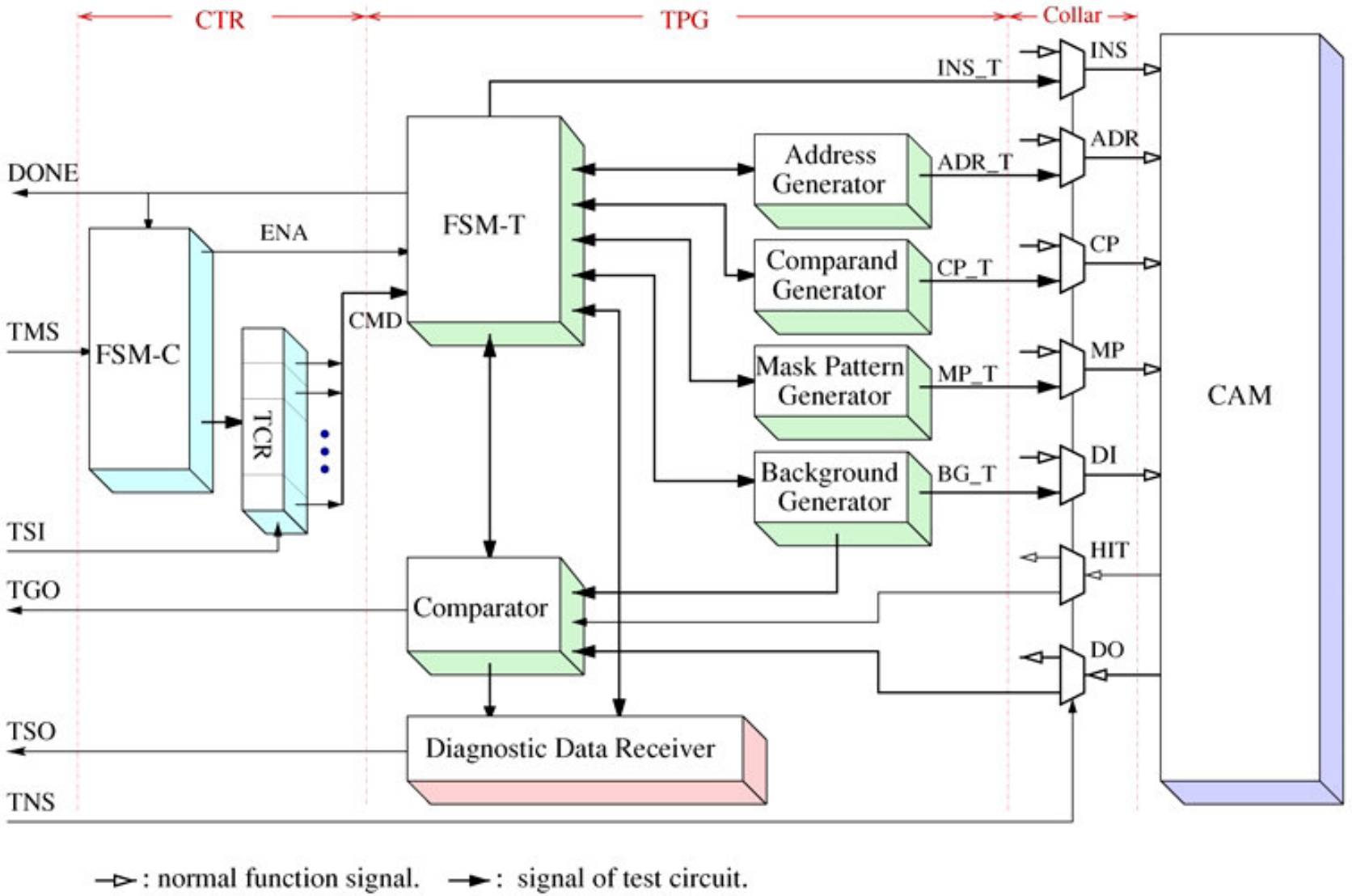
	SMF	SMMF	CMF	PMF	EMMF	IMF
T_{CAM}	100%	100%	100%	100%	100%	100%
NCDA	100%	100%	100%	100%	100%	100%
MLT-1	100%	100%	100%	100%	100%	100%

- Fault coverage for RAM faults

	SAF	TF	CF_{st}	CF_{id}	CF_{in}
T_{CAM}	100%	100%	90%	20%	40%
NCDA	100%	100%	90%	90%	100%
MLT-1	100%	100%	90%	90%	100%
MLT1+MLT-2	100%	100%	100%	100%	100%

[Source: J.-F. Li, R.-S. Tzeng and C.-W. Wu , JETTA03]

CAM BIST/BISD Implementation



[Source: J.-F. Li, R.-S. Tzeng and C.-W. Wu , JETTA03]

Testing Priority Encoder Faults of CAMs

[Source: J.-F. Li, ITC05]

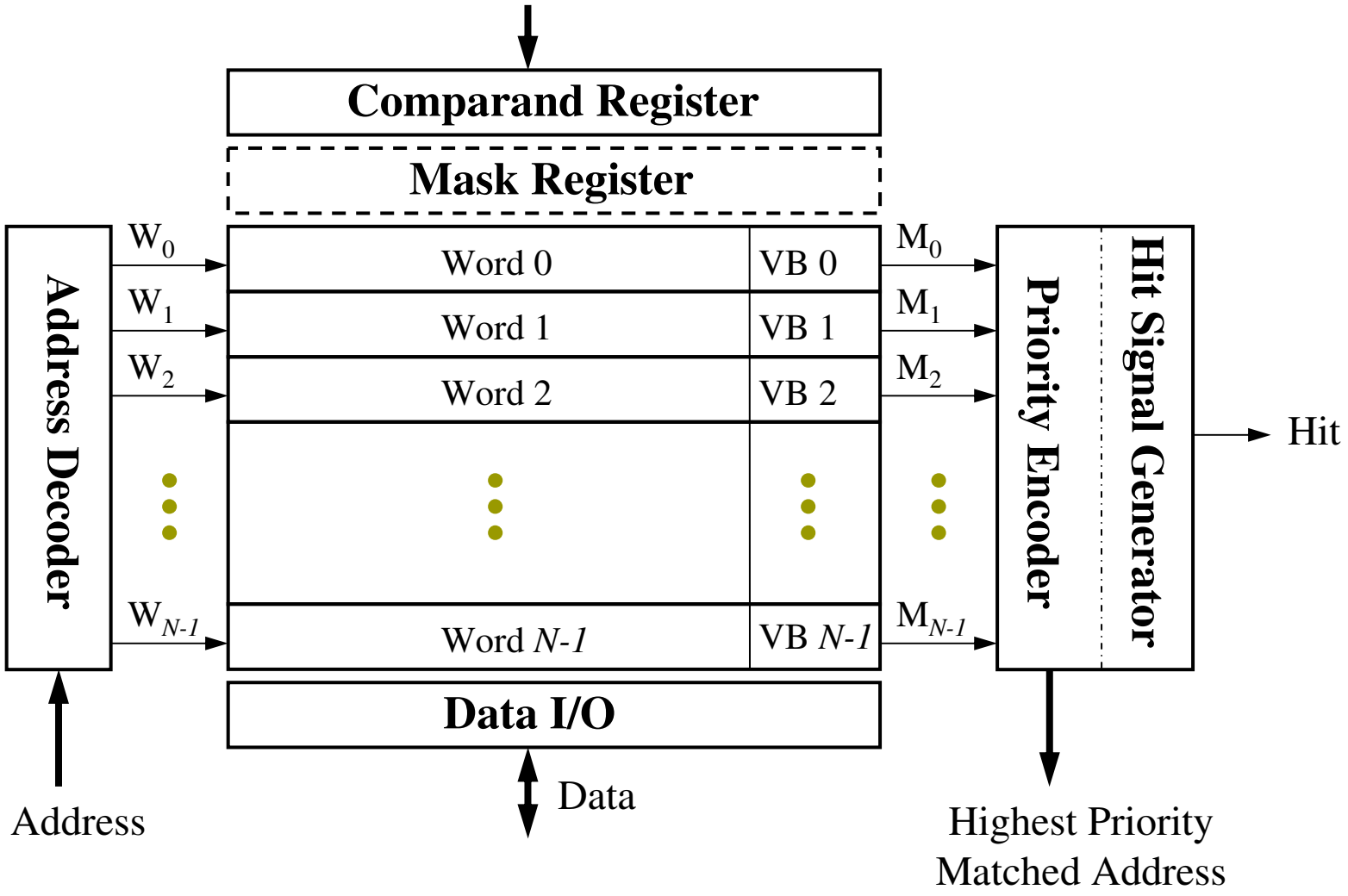
Previous Works on CAM Testing

- Testing of binary CAMs
  - P. Mazumder, et al., IEEE TCAD, 1988—test NPSFs in binary CAMs
  - K.-J. Lin and C.-W. Wu, IEEE TCAD00; P. R. Sidorowicz and J. A. Brzozowski, IEEE TCAD02—fault modeling & comparison fault testing
  - J. Zhao, et al., IEEE TC00—comparison and RAM fault testing for single-port and two-port CAMs
  - J.-F. Li, R.-S. Tseng, and C.-W. Wu, JETTA03—testing and diagnosis methodologies for BCAMs
  - J.-F. Li, IEICE Trans. Information & Systems, 2004—testing and diagnosing BCAMs with Comparison and RAM faults
- *All the previous works focus on testing cell array faults of CAMs*

Purpose

- *Show that conventional CAM tests for cell array faults cannot fully detect the SAFs in the priority encoder of CAMs*
- *Develop a test algorithm to detect the SAFs in the priority encoder of CAMs*

Typical CAM Architecture



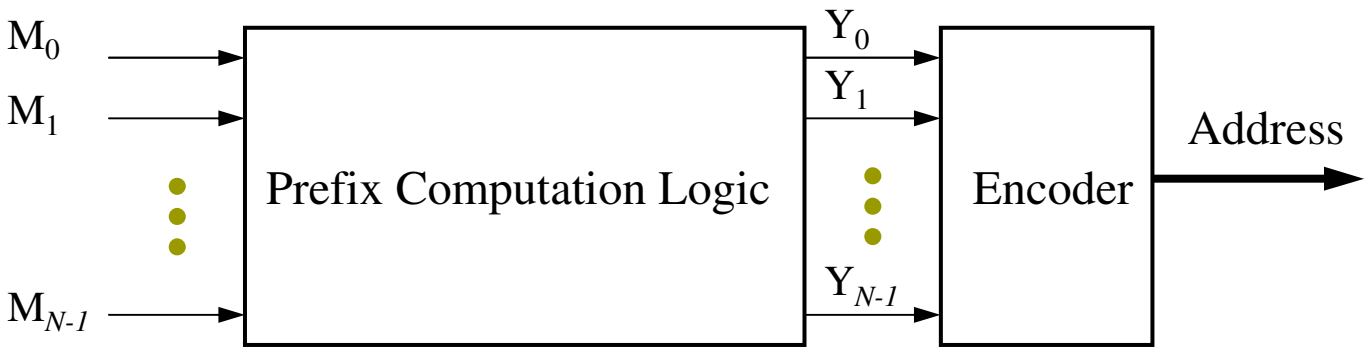
CAM Tests for Cell Array Faults

- Match signal responses when various CAM tests are applied

# Compare Operations	Expect Reponses ($M_0M_1…M_{N-1}$)	Type	Result Output
N	(100…000) (010…000) (001…000) … (000…001)	Type-1	Hit
N	(000…001) (000…011) (000…111) … (111…111)	Type-2	Lowest Matched Address
1	(000…000)	Type-3	Miss

Priority Encoder

- Architecture of the priority encoder

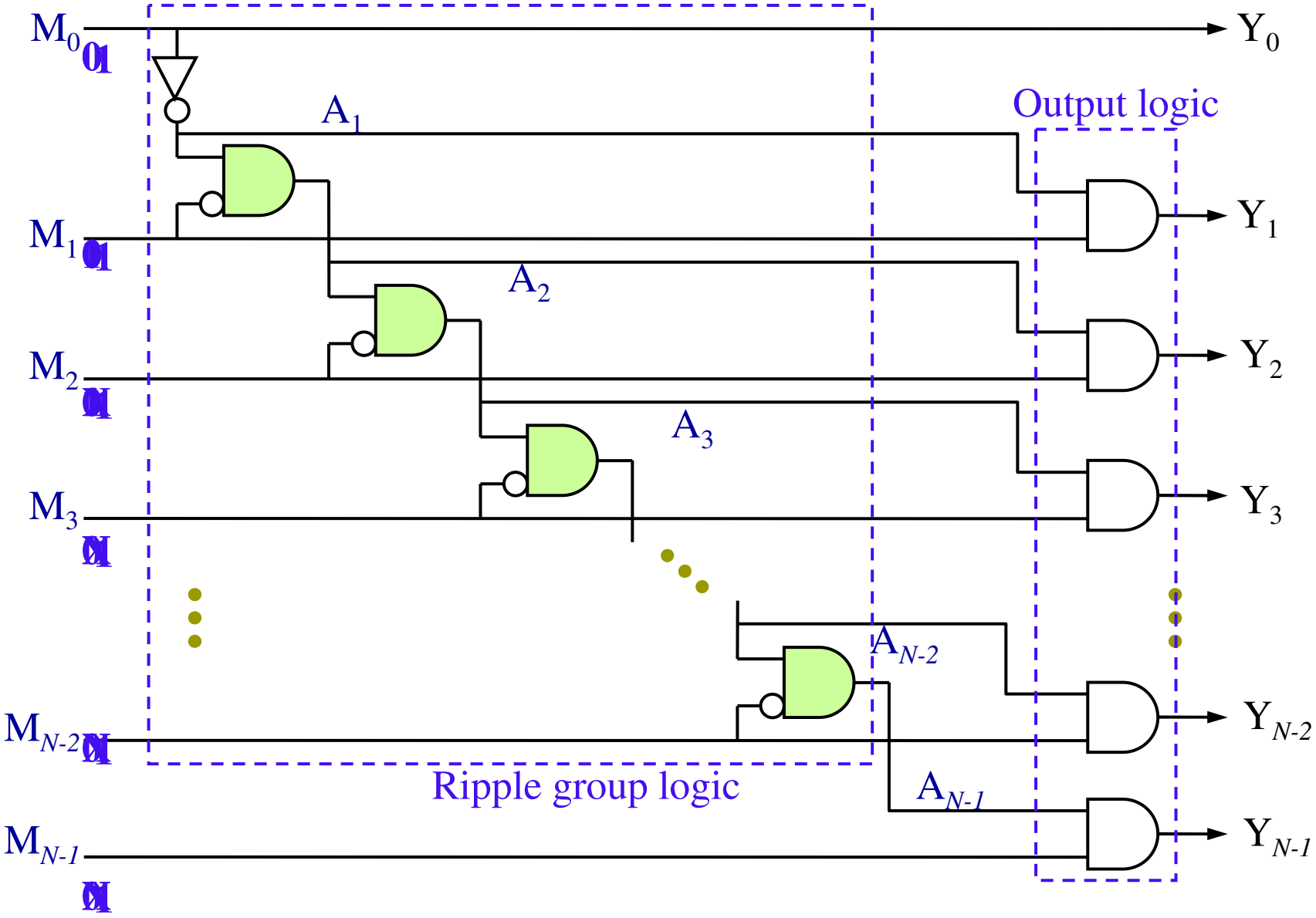


- Boolean function of the prefix computation logic
  - $Y_0 = M_0$
    $Y_1 = M_1 \overline{M}_0$
    $Y_2 = M_2 \overline{M}_1 \overline{M}_0$
    $\vdots$
    $Y_{N-1} = M_{N-1} \overline{M}_{N-2} \cdots \overline{M}_0$

Prefix Computation Logic (PCL)

- Let $A_i = \prod_{j=0}^{j=i-1} \overline{M}_j$, then $Y_0=M_0$ and $Y_i=A_iM_i$ for $1 \le i \le N-1$
- Thus the prefix computation logic usually can be realized by a two-stage logic circuit including a *group logic* ($Y_i=A_iM_i$) and a *output logic* ($A_i = \prod_{j=0}^{j=i-1} \overline{M}_j$) [N. Weste, 2005]
- Different group networks can be used to build the group logic
  - E.g., ripple, lookahead, tree, etc.
- We divide the testing of prefix computation logic into two parts
  - Testing of group logic
  - Testing of output logic

PCL with Ripple Group Logic

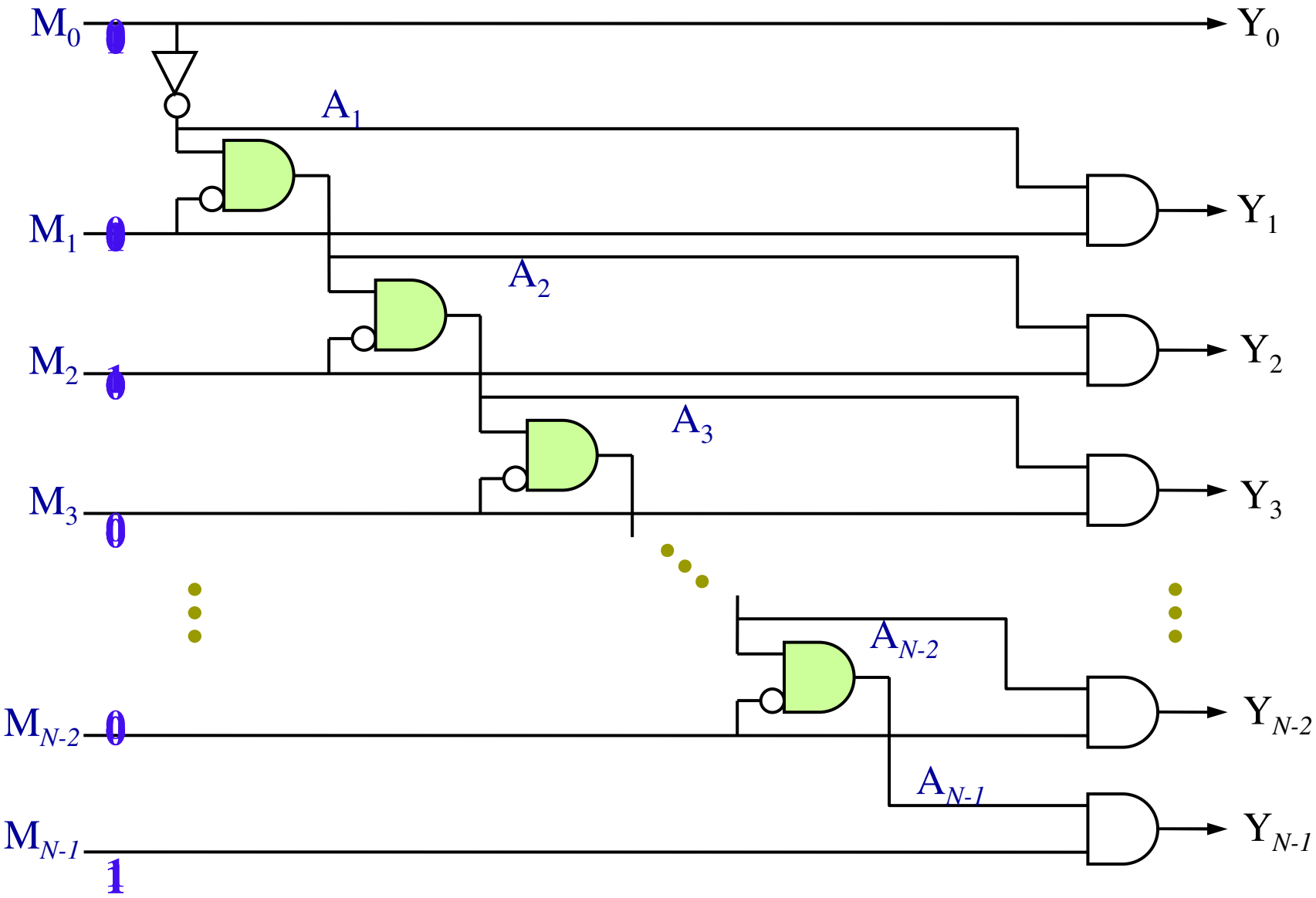


Testing of Output Logic

- Test patterns for detecting SAFs in output logic of the prefix computation logic

Test Pattern ($M_0M_1M_2....M_{N-1}$)	Test data received by AND_O_i
(1111…1111)	01 received by all AND_O_i
(0000…0000)	10 received by all AND_O_i
(01XX….XXXX)	11 received by AND_O_1
(001X….XXXX)	11 received by AND_O_2
(0001…XXXX)	11 received by AND_O_3
…	…
(0000…1XXX)	11 received by AND_O_{N-4}
(0000…01XX)	11 received by AND_O_{N-3}
(0000…001X)	11 received by AND_O_{N-2}
(0000…0001)	11 received by AND_O_{N-1}

Testing of Group Logic



Testing of Group Logic

- Test patterns for detecting SAFs in group logic of the prefix computation logic

Test Pattern ($M_0M_1M_2\ldots M_{N-1}$)	Test data received by AND_G_i
(0000…0001)	11 received by all AND_G_i
(1000….0001)	01 received by all AND_G_i
(0100….0001)	10 received by AND_G_1
(0010…0001)	10 received by AND_G_2
(0001…0001)	10 received by AND_G_3
…	…
(0000…1001)	10 received by AND_G_{N-4}
(0000…0101)	10 received by AND_G_{N-3}
(0000…0011)	10 received by AND_G_{N-2}

Testing SAFs of PCL

- Test patterns for detecting SAFs of PCL with ripple group logic (exporting the lowest matched address is assumed)

Test Pattern ($M_0M_1M_2\ldots M_{N-1}$)	Fault-free outputs of PCL ($Y_0Y_1Y_2\ldots Y_{N-1}$)
(1111…1111)	(1000…0000)
(0000…0000)	(0000…0000)
(0000…0001)	(0000…0001)
(1000….0001)	(1000…0000)
(0100….0001)	(0100…0000)
(0010…0001)	(0010…0000)
…	…
(0000…1001)	(0000…1000)
(0000…0101)	(0000…0100)
(0000…0011)	(0000…0010)

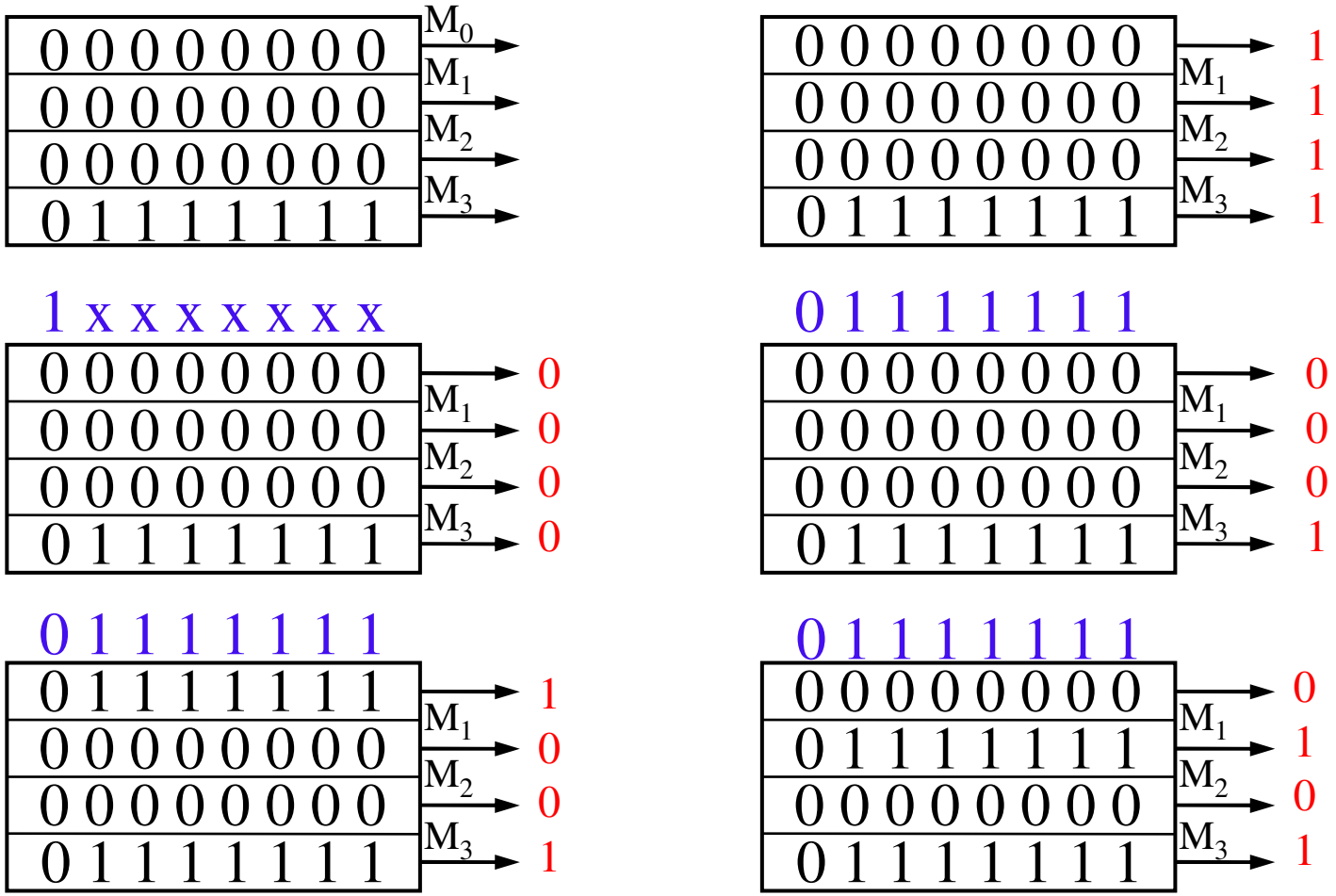
Proposed Test Algorithm

- Consider an NxB-bit CAM, the proposed test algorithm is as follows:

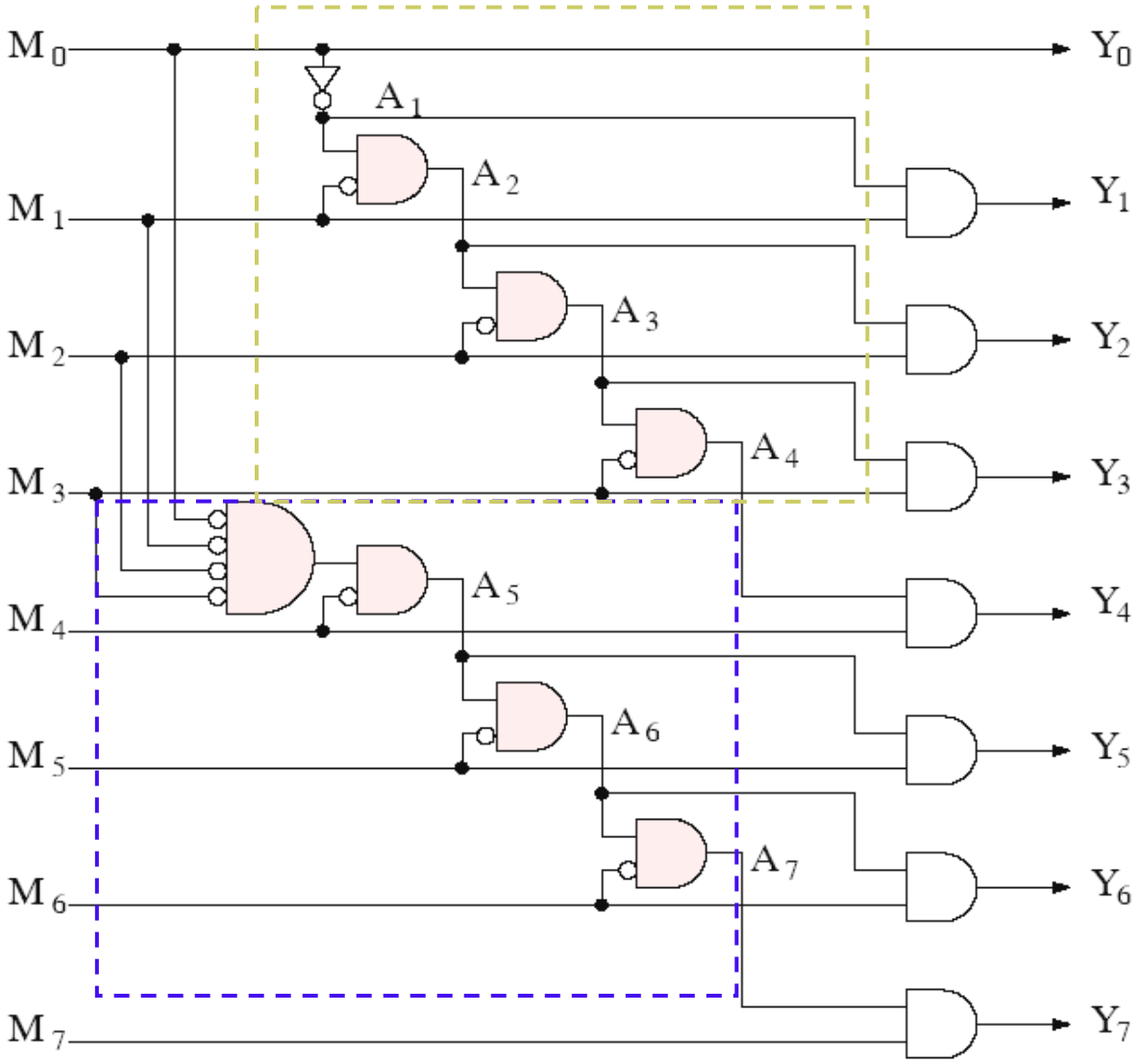
1) **FOR** $i=0$ to $N-1$ **DO** {
   IF ($i=N-1$) {
      Write on word i with the binary value of $2^{B-1}-1$;}
   ELSE {
      Write on word i with the binary value of 0;}}
2) Compare 0 with the bit $B-1$ of all words and the other bits of all words are masked. Check if the matched address is 0.
3) Compare 1 with the bit $B-1$ of all words and the other bits of all words are masked. Check if the matched address is invalid.
4) Compare $2^{B-1}-1$ with all words and check if the matched address is N.
5) **FOR** $i=0$ to $N-2$ **DO**{
   Write on word i with the binary value of $2^{B-1}-1$.
   Compare $2^{B-1}-1$ with all words and check if the matched address is i.
   Write on word i with the binary value of 0.}}

Example

- If a 4x8-bit CAM is tested, the test algorithm is applied as follows



Testing PCL with Lookahead Group Logic



Fault Coverage Analysis

- We use the Verifault of Verilog-XL simulator to simulate the fault coverage of SAFs of the prefix computation logic with ripple group logic
- Comparison of fault coverage

	Type-1, Type-3	Type-2, Type-3	T_{PE}
N=8	64.3%	91.4%	100%
N=16	62%	90.7%	100%
N=32	61%	90.3%	100%
N=64	60.5%	90.2%	100%

Comparison of Test Algorithms

- Let Test A (B) be a test algorithm causes that the CAM has Type-1 and Type-3 (Type-2 and Type-3) match signal responses
- Comparison of different test algorithms

	Priority Encoder Fault Coverage	Hit Signal Generator Fault coverage
Test A	Low	High
Test B	Medium	Low
Test A+T_{PE}	High	High
Test B+T_{PE}	High	Low

Conclusions

- Content addressable memories are widely used in digital systems, especially in network applications
- Efficient testing and diagnosing methodologies for CAMs are needed
- This lecture presents efficient testing and diagnosing methodologies for binary CAMs
- Also, a test algorithm for detecting the SAFs of priority encoder also has been presented