JOURNAL OF ELECTRONIC TESTING: Theory and Applications 19, 207–215, 2003

Testing and Diagnosis Methodologies for Embedded Content Addressable Memories

JIN-FU LI
Department of Electrical Engineering, National Central University, Jungli, Taiwan 320, R.O.C.

RUEY-SHING TZENG AND CHENG-WEN WU
Laboratory for Reliable Computing (LARC), Department of Electrical Engineering, National Tsing Hua University, Hsinchu, Taiwan 30013, R.O.C.

Received May 23, 2002; Revised November 21, 2002

Editor: A. Ivanov

Abstract. Embedded content addressable memories (CAMs) are important components in many system chips where most CAMs are customized and have wide words. This poses challenges on testing and diagnosis. In this paper two efficient March-like test algorithms are proposed first. In addition to typical RAM faults, they also cover CAM-specific *comparison* faults. The first algorithm requires $9N$ Read/Write operations and $2(N + W)$ Compare operations to cover comparison and RAM faults (but does not fully cover the intra-word coupling faults), for an $N \times W$-bit CAM. The second algorithm uses $3N \log_2 W$ Write and $2W \log_2 W$ Compare operations to cover the remaining intra-word coupling faults. Compared with the previous algorithms, the proposed algorithms have higher fault coverage and lower time complexity. Moreover, it can test the CAM even when its comparison result is observed only by the Hit output or the priority encoder output. We also present the algorithms that can locate the cells with comparison faults. Finally, a CAM BIST design is briefly described.

Keywords: BIST, CAM, march test algorithm, memory diagnostics, memory testing

1. Introduction

Content addressable memories (CAMs) play an important role in many digital systems and applications, such as computing, communication and networking, data compression, database management, cryptography, and signal processing. Recently, they are used in searching lookup tables, which is particularly important for packet-switched networks. With the network systems expected to deliver ever-higher bandwidth, the use of CAM will definitely keep growing. Most CAMs are embedded (and more or less customized), especially for system-on-chip (SOC) applications. They usually have very wide words, and sometimes can be observed only from the Hit output. For all these reasons, it is difficult to test the CAM cores using an external automatic test equipment (ATE) at a reasonable cost.

In this work we discuss only the static binary CAM, which is widely used in the industry. Each CAM cell consists of an SRAM cell and a comparison logic. This results in some differences between RAM and CAM testing. The CAM cell *comparison faults* need to be covered. Moreover, the observability of the effect of a comparison fault is usually low, since the fault effect has to be propagated to, e.g., the Hit output or the high-priority matched-address output. It thus is more difficult to locate the cells with comparison faults. Also, the built-in self-test (BIST) design for CAM is in general more complicated than that for RAM.

Several CAM BIST schemes have been reported before (see, e.g., [1, 3, 4, 6, 7]). The BIST scheme presented in [6] is used to detect stuck-at faults, adjacent cell coupling faults, and the neighborhood pattern sensitive faults (NPSFs) of dynamic CAMs. A design-for-testability (DFT) circuitry is added to the Hit signal generator to determine whether there is a hit for all the even match lines, and separately for all the odd match lines. In [7], a BIST circuit to test a reconfigurable CAM was proposed. The test circuit covers approximately 75% of the stuck-open faults and comparison faults in the cell array. In another work, the BIST circuit designed to test a cache memory (including the CAM and RAM blocks) [4] incorporates a serial interface into the cache such that it can execute a complete SMARCH algorithm [8] on the CAM Read/Write port. Yet another BIST architecture with a parallel test approach was reported in [3], which reduces the testing time by modifying the address decoder such that the parallel test approach can be used to detect coupling faults and NPSFs. Finally, in [1], a novel test algorithm was used to test a CAM whose priority encoder returns the lowest matching address. A fault effect can be masked if the fault occurs on a word whose address is higher than the matching address. A match compare circuit is added to each match line output such that the array can be fully tested by the BIST circuit.

In this paper, two March-like test algorithms are first proposed to cover the comparison faults and conventional RAM faults. The first algorithm can cover the comparison and RAM faults (excluding intra-word coupling faults) with $9N$ Read/Write and $2(N+W)$ Compare operations for an $N \times W$-bit CAM. The second algorithm detects the intra-word coupling faults using $3N \log_2 W$ Write and $2W \log_2 W$ Compare operations. In comparison with the previous test algorithms, the proposed algorithms have a higher fault coverage and lower time complexity. Moreover, they are good for many kinds of CAMs, e.g., the one whose comparison result is observed by the Hit output only or the priority encoder output only. Fault-location algorithms are also proposed. They are for locating the cells with comparison faults. A CAM BIST circuit design is also briefly discussed. The BIST design supports a wide range of test algorithms, and is good for various fault coverage requirements. Since diagnostics also is considered, both the production test and engineering/diagnostic test can be done.

The paper is organized as follows. Section 2 introduces the CAM architecture and functional faults. The March-like test and fault-location algorithms are presented in Section 3. Section 4 summarizes the experimental results and gives the comparison. A BIST design is also briefly described. Finally, Section 5 concludes the paper.

2. CAM Architecture and Fault Models

2.1. CAM Architecture

The architecture of a typical CAM is depicted in Fig. 1. The Address Decoder and Data I/O blocks are similar to those in a RAM. The Mask Register stores a binary pattern determining whether the corresponding bits in a word are to be masked from further Write and Compare operations or not. Each of the Valid Bit Flip-Flops indicates whether the match signal of the corresponding word is valid or invalid. The Hit Signal Generator evaluates the valid match signals, and generates a hit output (Hit = 1) if there is a valid match. The Priority Encoder outputs the address of the highest priority word that are matched. Note that the Priority Encoder and Hit Signal Generator may not co-exist.

Fig. 2 shows an $N \times W$ CAM cell array, where N is the number of words and W is the number of bits within a word. Each cell is composed of an SRAM cell and a comparison logic (formed by transistors T3, T4, and T5) [9]. A CAM usually has the following basic operations: Write, Read, Compare, Erase, and Masked Compare. The Read and Write operations are the same as those of a RAM. The Masked Compare operation compares an input pattern with all words in the CAM simultaneously, with one or more bits blocked (not compared) by setting the corresponding bits of the mask pattern. The Masked Write operation writes an

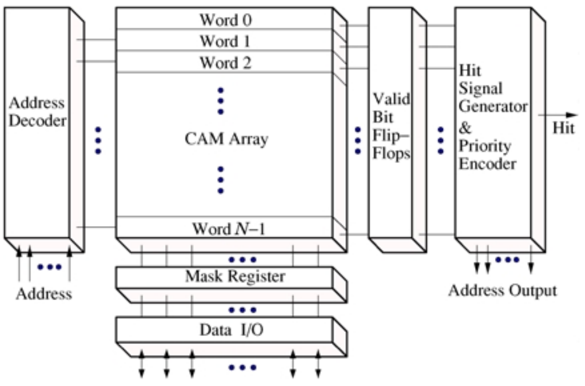

Fig. 1. Architecture of a typical CAM.

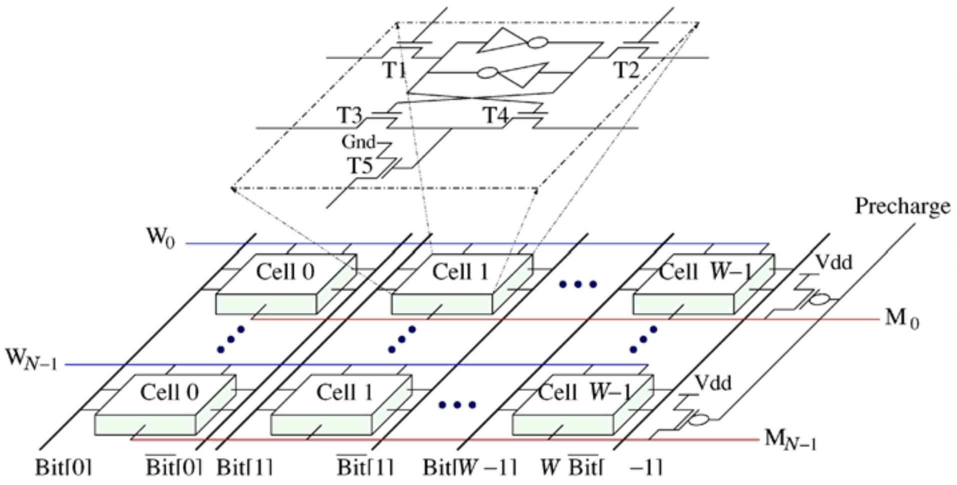

Fig. 2. An $N \times W$ CAM array and the cell structure.

input pattern to a specified word, with one or more bits blocked (not written) by setting the corresponding bits of the mask pattern. The Write operations also set the valid bit of the corresponding word (so that it is in the *valid state*). The word-line pass transistors (T1 and T2) are turned off when the CAM executes the Compare and Masked Compare operations. The match lines (M_i) are precharged to Vdd before the Compare operation, by resetting the Precharge signal. The input pattern is then compared with the all the CAM words simultaneously. If there is any word storing a pattern the same as the input pattern, the corresponding match signal will be high (1) since all the T5 transistors of the word are turned off. Also, the Hit signal will be 1. Finally, the Erase operation resets the corresponding Valid Bit Flip-Flop of a specified word.

2.2. *Functional Fault Models*

As discussed above, a CAM cell consists of two parts: one is the storage unit that is the same as an SRAM cell, and the other is the comparison logic. The faults thus can be classified into two classes: RAM faults and comparison faults. A reduced functional model for RAM consists of three blocks: the address decoder, the memory cell array, and the Read/Write logic [10]. Typical RAM faults [2, 10] are described as follows: (1) single-cell faults: stuck-at fault (SAF)—the cell contains a 0, i.e., SAF(0), or a 1, i.e., SAF(1), and cannot be changed; transition fault (TF)—a cell can not undergo a 0 to 1 transition (i.e., $\uparrow$) or a 1 to 0 transition (i.e., $\downarrow$); (2) two-cell faults: state-coupling fault (CFst)—the coupled cell (or victim) is forced to 0 or 1 only if the coupling cell (or aggressor) contains 0 or 1; idempotent coupling fault (CFid)—the coupled cell is forced to 0 or 1 if the coupling cell is given a $\uparrow$ or $\downarrow$.

The comparison faults [5] are described as follows: (1) stuck-matched fault (SMF)—a cell always matches the corresponding input bit irrespective of the CAM cell state and input pattern; (2) stuck-mismatched fault (SMMF)—there is no match for the cell irrespective of the CAM cell state and input pattern; (3) partial-match fault (PMF)—a CAM cell is stuck-matched for all subsequent Compare operations when a logic value x (either 0 or 1) is written into the cell, and stuck-mismatched when $\bar{x}$ is written into it; (4) conditional-match fault (CMF)—a cell functions correctly if it stores a logic value x, but it always provides an incorrect result for the subsequent Compare operations if it stores $\bar{x}$; (5) equivalence-mismatch fault (EMMF)—the Compare operation fails if the CAM cell stores a value x and is compared with the same input value x; (6) inequivalence-match fault (IMF)—the Compare operation fails if the CAM cell stores a value x and is compared with the complementary input value $\bar{x}$; (7) cross-match fault (XMF) and cross-mismatch fault (XMMF)—caused by a short between two neighboring *bit* and $\overline{bit}$ lines belonging to two neighboring cells in the same word, abnormal matching or mismatching occurs in one cell or the other (see the details in [5]); (8) stuck-valid fault (SVF)—a word always participates in the Compare operation even if it has been invalidated by a previous erase operation; (9) stuck-invalid fault (SIVF)—a word cannot be validated, i.e.,

Table 1. Cell responses to all Compare-after-Write operations under various comparison faults [5].

Operation	SMF	SMMF	CM1F	CM0F	PM1F	PM0F	EMM1F	EMM0F	IM1F	IM0F
$w0, c0$	M	MM	MM	M	MM	M	M	MM	M	M
$w0, c1$	M	MM	M	MM	MM	M	MM	MM	MM	M
$w1, c0$	M	MM	MM	M	M	MM	MM	MM	M	MM
$w1, c1$	M	MM	M	MM	M	MM	MM	M	M	M

it never participates in any Compare operation even if it has been written previously; and (10) Mask Register stuck-at fault (MSAF)—one or more Mask Register cells are separately stuck-at 0 or 1.

Table 1 summarizes the cell responses to all Compare-after-Write operations under various single cell comparison faults [5]. In the table, "M" and "MM" denote the "match" and "mismatch" results of the corresponding $(wx; cy)$ operations, respectively, where $x, y \in \{0, 1\}$. For example, if a cell has a conditional-match-zero fault (CM0F), then the behavior of the faulty cell is as follows. If a 0 is stored in the cell, then the cell has a correct Compare operation, but its Compare result is incorrect if the cell stores a 1. Possible defects causing these comparison faults are reported in [5].

3. March-Like Algorithms

3.1. Test Algorithms

We follow the notation and definitions given in [5, 10]. A test *algorithm* (or simply *test*) contains a sequence of one or more *test elements*. Each test element consists of a number of CAM operations, possibly with a prespecified address sequence, which can be ascending ($\Uparrow$), descending ($\Downarrow$), or either way ($\Updownarrow$). The compared patterns can be masked or non-masked patterns. A masked pattern is obtained from an input pattern filtered by the value of the Mask Register. The CAM operations are: (1) wD: write an input pattern D (a binary word, with **0** and **1** indicating the all-0 and all-1 patterns, respectively); (2) rD: read an expected data D; (3) cP_D^M: given the mask pattern M, compare an input pattern D with all words in the CAM, where the M can be omitted if no masking is needed; and (4) E (Erase): reset the valid bit of a specified word, i.e., invalidate the word.

We now present two test algorithms that cover the comparison and RAM faults (including inter-word and intra-word CFs). Let $\omega(i) = 2^W - 1 - 2^i$, where W is the width of a word. The first March-like test (MLT-1) is described as follows:

$$\text{MLT-1} = \big\{\Updownarrow(w\mathbf{1}); \Uparrow(w\mathbf{0}, cP_{\mathbf{0}}; w\mathbf{1}); \Uparrow(r\mathbf{1}, w\mathbf{0}); \big(cP_{\mathbf{1}}^{\omega(0)}, cP_{\mathbf{1}}^{\omega(1)}, \ldots, cP_{\mathbf{1}}^{\omega(W-1)}\big); \Downarrow(w\mathbf{1}, cP_{\mathbf{1}}, w\mathbf{0}); \Downarrow(r\mathbf{0}, w\mathbf{1}); \big(cP_{\mathbf{0}}^{\omega(0)}, cP_{\mathbf{0}}^{\omega(1)}, \ldots, cP_{\mathbf{0}}^{\omega(W-1)}\big)\big\}$$

Note that each Compare operation within the 4th and 7th test elements is performed on all words of a CAM, so no prespecified address sequence is needed. Also, the number of test operations depends on the width of a word.

In MLT-1, the first test element initializes the CAM array to the all-1 solid background. Table 2 shows the fault-free status of a 3×3 CAM when the second test element is executed. The first word is denoted as Word 0 (W_0), and the operations $w\mathbf{0}$, $cP_{\mathbf{0}}$, and $w\mathbf{1}$ are executed at W_0 in sequence. The $w\mathbf{0}$ operation writes an all-0 pattern into W_0. Note that the match signals are all in don't-care (x) state during the Write or Read operation. The $cP_{\mathbf{0}}$ operation compares the all-0 pattern with the contents of all words (i.e., W_0, W_1, and W_2) in the CAM concurrently. If W_0 is fault free, then $M_0 = 1$, and Hit $= 1$. If, e.g., W_0 has a faulty comparison logic or wrong state, then $M_0 = 0$ after the $cP_{\mathbf{0}}$ operation. Therefore, Hit $= 0$ and the fault can be

Table 2. Fault-free status when element $\Uparrow(w\mathbf{0}, cP_{\mathbf{0}}, w\mathbf{1})$ is executed.

		Content (M_i)		
Operation		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)

Table 3. Fault-free status when element $\Uparrow(r\mathbf{1}, w\mathbf{0})$ 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)

detected. As another example, if the second cell of W_0 is stuck-at 1, then $M_0 = 0$, and Hit $= 0$. The operations are repeated at W_1, and finally at W_2. In summary, this test element detects the stuck-at faults and inter-word 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, XMMF, and SIVF can be detected since they cause the match signals to become mismatched or invalid, and the Hit output is 0.

Table 3 shows the fault-free status of the CAM when the third test element is executed. The $w\mathbf{0}$ and $w\mathbf{1}$ operations in the second test element have caused all the cells to undergo both the $\uparrow$ and $\downarrow$ transitions. Therefore, the $r\mathbf{1}$ operation here can detect the inter-word CFid where the aggressors are located at higher addresses than the victims. Also, the cells located at lower addresses than the current word will undergo a $\downarrow$ transition after the $w\mathbf{0}$ operation. The Read operation thus also can detect the inter-word CFid where the aggressors are located at lower addresses than the victims. Finally, 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 can also be detected.

Table 4 shows the fault-free status of the CAM when the fourth test element is executed. The Compare operations are masked, i.e., the operation $cP_{\mathbf{1}}^{\omega(i)}$ compares the ith bit in all CAM words with the value 1. For example, in $cP_{\mathbf{1}}^{6}$, then the column 0 of the CAM is compared with 1, since $6 = 110$ in binary and columns 1 and 2 are masked. Hit $= 0$ if there is no 1 in that column, as in the fault-free case. If, e.g., there is an SMF, then Hit $= 1$ and the fault is detected. After the entire test element is finished, any of the SMF, CMF, and IMF can be detected. 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. By a similar approach, the faults detected by other test elements can be derived. The fault coverage of MLT-1 will be discussed in Section 4.

Table 4. Fault-free status when element $(cP_{\mathbf{1}}^{\omega(0)}, cP_{\mathbf{1}}^{\omega(1)}, \ldots, cP_{\mathbf{1}}^{\omega(W-1)})$ is executed.

	Content (M_i)		
	$cP_{\mathbf{1}}^{6}$	$cP_{\mathbf{1}}^{5}$	$cP_{\mathbf{1}}^{3}$
W_0	000 (0)	000 (0)	000 (0)
W_1	000 (0)	000 (0)	000 (0)
W_2	000 (0)	000 (0)	000 (0)

Note that the intra-word CFst and CFid are not fully covered by MLT-1, since only the solid (all-0 and all-1) backgrounds are used. A short test algorithm called March-CW [11] was proposed to cover the intra-word CFs for RAM, whose key element is $\Updownarrow (wD, w\bar{D}, r\bar{D}, wD, rD)$, where $D = \{0101, 0011\}$ for 4-bit words. However, March-CW is not the most efficient for CAM so far as test time is concerned. We propose the MLT-2 test algorithm for the intra-word CFs as shown below, which is more efficient.

$$\begin{aligned}\text{MLT-2} = \{ & \Updownarrow (wD); \Updownarrow (w\bar{D}); \\ & (cP_D^{\omega(0)}, cP_D^{\omega(1)}, \ldots, cP_D^{\omega(W-1)}); \\ & \Updownarrow (wD); (cP_{\bar{D}}^{\omega(0)}, cP_{\bar{D}}^{\omega(1)}, \ldots, cP_{\bar{D}}^{\omega(W-1)}) \}\end{aligned}$$

In MLT-2 the Write operations, if supplied by different data backgrounds as in March-CW, can activate different intra-word CFs. The Compare operations are used to observe faults. If there is an intra-word CF, then the state of the faulty cell is changed after one of the Write operations. The third and fifth test elements allow us to observe the faults. Hit $= 0$ if and only if the CAM is fault free (with respect to intra-word CF). The complexity is $3N \log_2 W$ Write operations and $2W \log_2 W$ Compare operations for MLT-2, as opposed to $3N \log_2 W$ Write operations and $2N \log_2 W$ Read operations for March-CW. Clearly, MLT-2 is faster if N is larger than W, which is true in most cases.

3.2. Diagnostic Algorithms

As discussed in Section 3.1, the comparison faults can be detected by the Compare operations. However, we only know the faulty word or column, but the faulty cells in the word or the corresponding columns in the array can not be identified directly. We propose the following fault location algorithms for CAM fault

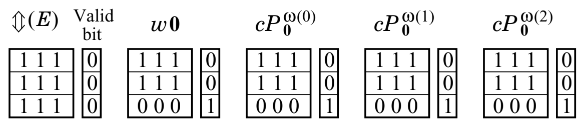
$\Updownarrow(E)$
Valid bit
$w\mathbf{0}$
$cP_{\mathbf{0}}^{\omega(0)}$
$cP_{\mathbf{0}}^{\omega(1)}$
$cP_{\mathbf{0}}^{\omega(2)}$
111 0
111 0
111 0
111 0
111 0
000 1

Fig. 3. An example for locating the faulty cells in W_2.

diagnostics. The faulty cells in a word (row) can be located by one of the two algorithms:

$$\text{FLR-0} = \left\{\Updownarrow(E); w\mathbf{0}; \left(cP_{\mathbf{0}}^{\omega(0)}, cP_{\mathbf{0}}^{\omega(1)}, \ldots, cP_{\mathbf{0}}^{\omega(W-1)}\right)\right\}$$
$$\text{FLR-1} = \left\{\Updownarrow(E); w\mathbf{1}; \left(cP_{\mathbf{1}}^{\omega(0)}, cP_{\mathbf{1}}^{\omega(1)}, \ldots, cP_{\mathbf{1}}^{\omega(W-1)}\right)\right\}$$

For example, assume that the test element $\Uparrow (w\mathbf{0}, cP_{\mathbf{0}}, w\mathbf{1})$ of MLT-1 detects a fault in word W_2 of the 3×3 CAM described previously. Then, FLR-0 can be used to locate the faulty cells in W_2. Fig. 3 shows an example for the fault-location process. The test element $\Updownarrow$ (E) first resets all valid bits in either the ascending or descending address sequence. The $w\mathbf{0}$ operation on W_2 then sets its valid bit. The subsequent Compare operations thus are valid only for W_2. Note that the $w\mathbf{0}$ operation does not change the state of the faulty cells (victims) since other words are not changed. The third test element of FLR-0 can locate the faulty cells by performing the Compare operation bit by bit. Similarly, if the test element $\Downarrow (w\mathbf{1}, cP_{\mathbf{1}}, w\mathbf{0})$ of MLT-1 detects a faulty word, then FLR-1 can be used to locate the faulty cells in the word.

The proposed algorithms for locating the faulty cells in a column (the same bit in all words) are as follows.

$$\text{FLC-0} = \left\{ \Updownarrow (E); \Updownarrow \left(w\mathbf{0}, cP_{\mathbf{1}}^{M}, E\right)\right\}$$
$$\text{FLC-1} = \left\{ \Updownarrow (E); \Updownarrow \left(w\mathbf{1}, cP_{\mathbf{0}}^{M}, E\right)\right\}$$

For example, if the Compare operation $cP_{\mathbf{1}}^{\omega(0)}$ in the 4th test element of MLT-1 detects a fault in the 3×3 CAM, then we know that column 0 is faulty. Then, FLC-0 can be used to identify the faulty bits of the column, with $M = \omega(0)$ (i.e., 110 in binary). Note that M is the same as the mask pattern of the Compare operation detecting the faulty column. Fig. 4 shows an example for the fault-location process. The Erase operation

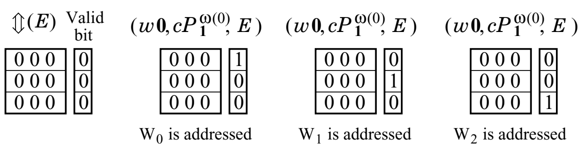
$\Updownarrow(E)$
Valid bit
$(w\mathbf{0}, cP_{\mathbf{1}}^{\omega(0)}, E)$
$(w\mathbf{0}, cP_{\mathbf{1}}^{\omega(0)}, E)$
$(w\mathbf{0}, cP_{\mathbf{1}}^{\omega(0)}, E)$
000 0
000 0
000 0
000 1
000 0
000 0
000 0
000 1
000 0
000 0
000 0
000 1
W_0 is addressed
W_1 is addressed
W_2 is addressed

Fig. 4. An example for locating the faulty cells in column 0.

first resets all valid bits. The $w\mathbf{0}$ operation then writes an all-0 pattern to W_0 (the ascending address sequence is assumed) and sets its valid bit. Now the input pattern is only compared with bit 0 of W_0, and it will be identified if it is faulty. The Erase operation resets the valid bit of W_0 subsequently, so all the faulty cells in column 0 can be identified after FLC-0 is finished. Similarly, FLC-1 can be used to locate the faulty cells in the faulty column detected by the 7th test element of MLT-1.

4. Experimental Results

For comparison, in Table 5 we show the fault coverage numbers from three CAM test algorithms: T_{CAM} [5], NCDA [12], and the proposed MLT-1. The fault coverage numbers are reported by the CAM fault simulator that we have developed, which was extended from our RAM fault simulator, RAMSES [11]. In the table, the results for T_{CAM}, NCDA, and MLT-1 are based on the same assumption that the comparison result is observed only by the priority encoder output, while the results for T^*_{CAM}, NCDA*, and MLT-1* are based on the same assumption that the comparison result is observed only by the Hit output. From the table, both T_{CAM} and T^*_{CAM} cover all the comparison faults; while MLT-1, MLT-1*, NCDA, and NCDA* do not detect XMF, XVF, or MSAF. Moreover, by NCDA* the coverage for SMMF, EMMF, and XMMF is only 1/N%, since only the faults in the first addressed word can be detected and the others cannot be covered due to fault masking. Note that in this paper we do not provide algorithms for detecting XMF, XVF, and MSAF, since they are covered by the existing efficient test algorithms T_{XM}, T_{SV} , and T_{SU}, respectively [5].

Table 6 shows the fault coverage numbers for the three algorithms, with respect to the conventional RAM faults. The results show that NCAD and MLT-1 has 90% coverage of CFst and CFid (assuming $W = 4$). The reason is that some intra-word CFst and CFid are not covered. If MLT-2 is added, then they can be fully covered. Note that T_{CAM} has lower coverage for CFid, since only solid backgrounds are used. Also, there is no Read operations in the algorithm.

The time complexities of the three algorithms, for an $N \times W$ CAM, are as follows: T_{CAM} requires $8N$ Write operations, $2N + 3W + 2$ Compare operations, and N Erase operations; NCDA requires $5N$ Write operations, $5N$ Read operations, and $2(N + W)$ Compare operations; and MLT-1 uses $7N$ Write operations, $2N$ Read operations, and $2(N + W)$ Compare operations.

Table 5. Fault coverage comparison for three test algorithms.

	SMF	SMMF	CMF	PMF	EMMF	IMF	XMMF	XMF	SIVF	SVF	MSAF
T_{CAM}	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
NCDA	100%	100%	100%	100%	100%	100%	100%	0%	100%	0%	0%
MLT-1	100%	100%	100%	100%	100%	100%	100%	0%	100%	0%	0%
T_{CAM}*	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
NCDA*	100%	$1/N\%$	100%	100%	$1/N\%$	100%	$1/N\%$	0%	100%	0%	0%
MLT-1*	100%	100%	100%	100%	100%	100%	100%	0%	100%	0%	0%

Table 6. Fault coverage statistics for RAM faults.

	SAF	TF	CFst	CFid
T_{CAM}	100%	100%	90%	20%
NCDA	100%	100%	90%	90%
MLT-1	100%	100%	90%	90%
MLT-1 + MLT-2	100%	100%	100%	100%

In general, MLT-1 has higher fault coverage and lower time compexity as compared with NCDA. Also, though T_{CAM} is the best choice for comparison faults, its BIST implementation is quite complicated, so unless XMF, SVF, and MSAF are among the faults that must be covered, MLT-1 is better for BIST implementation.

We also have developed a CAM BIST scheme. From the above discussion, so far no single algorithm covers all the comparison and RAM faults. The hardware overhead would be quite high if the BIST circuit has a built-in algorithm that covers all these faults, so programmability is very important for CAM BIST. In our BIST scheme, differnt test algorithms can be applied. For example, the test algorithms T_{XM}, T_{SV} , and T_{SU} reported in [5] for detecting XMF, SVF, and MSAF are supported. Diagnostic algorithms also can be applied by using our BIST circuit. Programmability and diagnostic support are two major features of our BIST design. Fig. 5 depicts the block diagram of the proposed BIST design, which is composed of the Controller (CTR), Test Pattern Generator (TPG), and Collar. The

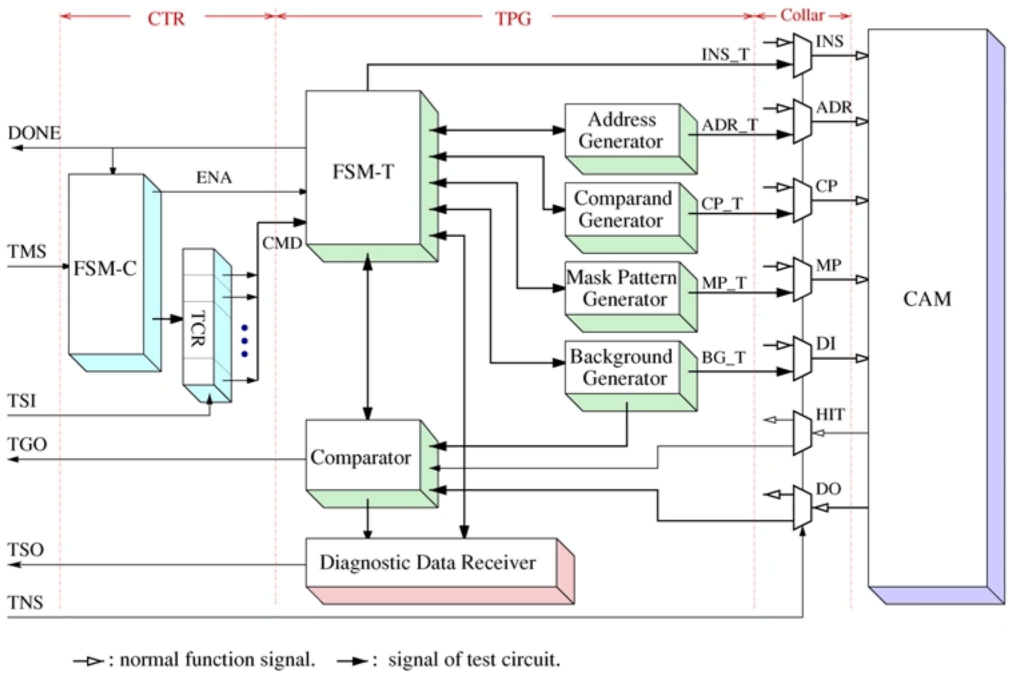

Fig. 5. Block diagram of the proposed BIST design.

CTR issues test commands for the TPG that generates test patterns to the CAM and handles the response. The TPG is different from that for a RAM—it contains the Comparand Generator and Mask Pattern Generator in addition to the Background Generator, Comparator, and Diagnostic Data Receiver.

To evaluate the hardware cost of the proposed BIST scheme, we have modeled the BIST design using various CAM configurations at the synthesizable RTL level, and predicted the area overhead figures by a synthesis tool. The area overhead was reported in terms of gate count, using a 0.35 μm CMOS cell library. For example, for $N = 2^{13}$ (i.e., an 8K-word CAM), the BIST gate counts are 2,540 for 64-bit words, respectively. The numbers are small and acceptable in most cases.

5. Conclusion

In this paper we have presented efficient testing and fault-location algorithms and a programmable BIST scheme for CAM. Both CAM-specific comparison faults and conventional RAM faults are considered. The first algorithm (MLT-1) requires $9N$ Read/Write operations and $2(N + W)$ Compare operations to detect the comparison and RAM faults (except intra-word CFs) for and $N \times W$-bit CAM. The second algorithm (MLT-2) covers the intra-word CFs with $3N \log_2 W$ Write operations and $2W \log_2 W$ Compare operations. Experimental results show that the proposed testing algorithms have high fault coverage and low time complexity. Moreover, it can test a CAM even when its comparison result is observed only by the Hit output or the priority encoder output. A shorter test algorithm with $6 \log_2 W$ operations is proposed to test the faults existed in peripheral circuitry. Fault-location algorithms also have been proposed that can locate the cells with comparison faults. The proposed programmable BIST supports different algorithms, and it also has diagnostic support. The hardware overhead is small—about 2,540 gates for an 8k $\times$ 64-bit CAM.

In this work, the diagnostic algorithms are only used to identify the positions of the faulty cells. Our future work is to investigate the diagnostic algorithms that can also distinguish between different fault types.

References

1. T. Chadwick, T. Gordon, R. Nadkarni, and J. Rowland, "An ASIC-Embedded Content Addressable Memory with Power-Saving and Design for Test Features," in *Proc. IEEE Custom Integrated Circuits Conf. (CICC)*, 2001, pp. 183–186.
2. R. Dekker, F. Beenker, and L. Thijssen, "A Realistic Fault Model and Test Algorithm for Static Random Access Memories," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 9, no. 6, pp. 567–572, June 1990.
3. Y.S. Kang, J.C. Lee, and S. Kang, "Parallel BIST Architecture for CAMs," *Electronics Letters*, vol. 33, no. 1, pp. 30–31, Jan. 1997.
4. S. Kornachuk, L. McNaughton, R. Gibbins, and B. NadeauDostie, "A High Speed Embedded Cache Design with Non-Intrusive BIST," in *Proc. IEEE Int. Workshop on Memory Technology, Design and Testing (MTDT)*, San Jose, 1994, pp. 40–45.
5. K.-J. Lin and C.-W. Wu, "Testing Content-Addressable Memories Using Functional Fault Models and March-Like Algorithms," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 5, pp. 577–588, May 2000.
6. P. Mazumder, J.H. Patel, and W.K. Fuchs, "Methodologies for Testing Embedded Content Addressable Memories," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 7, no. 1, pp. 11–20, Jan. 1988.
7. A.J. McAuley and C.J. Cotton, "A Self-Testing Reconfigurable CAM," *IEEE Journal of Solid-State Circuits*, vol. 26, no. 3, pp. 257–261, March 1991.
8. B. Nadeau-Dostie, A. Silburt, and V.K. Agarwal, "A Serial Interfacing Technique for External and Built-in Self-Testing of Embedded Memories," *IEEE Design & Test of Computers*, vol. 7, no. 2, pp. 56–64, April 1990.
9. K.J. Schultz, "Content-Addressable Memory Core Cells: A Survey," *Integration, the VLSI J.*, vol. 23, pp. 171–188, 1997.
10. A.J. van de Goor, *Testing Semiconductor Memories: Theory and Practice*, Chichester, England: John Wiley & Sons, 1991.
11. C.-F. Wu, C.-T. Huang, and C.-W. Wu, "RAMSES: A Fast Memory Fault Simulator," in *Proc. IEEE Int. Symp. Defect and Fault Tolerance in VLSI Systems (DFT)*, Albuquerque, Nov. 1999, pp. 165–173.
12. J. Zhao, S. Irrinki, M. Puri, and F. Lombardi, "Testing SRAM-Based Content Addressable Memories," *IEEE Trans. Computers*, vol. 49, no. 10, pp. 1054–1063, Oct. 2000.

Jin-Fu Li received the BSEE degree in 1995 from National Taiwan University of Science Technology, Taipei, Taiwan, and the MSEE and Ph.D. degrees, both in Electrical Engineering, in 1999 and 2002, respectively, from National Tsing Hua University, Hsinchu, Taiwan. Since 2002 he has been with the Department of Electrical Engineering, National Central University, Chung-Li, Taiwan, where he is currently an assistant professor.

His research interests include advanced VLSI design and testing, SOC testing, SOC reliable design, memory testing, memory diagnosis, and memory self-repair.

Ruey-Shing Tzeng received the BSEE and MSEE degrees in 1996 and 2001, respectively, from National Tsing Hua University, Hsinchu, Taiwan. He is currently with MediaTek Inc., Hsinchu, Taiwan, as an IC design engineer. His research interests include memory testing, memory diagnosis, and memory self-repair.

Cheng-Wen Wu received the BSEE degree in 1981 from National Taiwan University, Taipei, Taiwan, and the MS and Ph.D. degrees in electrical and computer engineering, in 1985 and 1987, respectively, from the University of California, Santa Barbara. Since 1988 he has been with the Department of Electrical Engineering, National Tsing Hua University (NTHU), Hsinchu, Taiwan, where he is currently a professor. He also has served as the director of the University's Computer and Communications Center from 1996 to 1998, and the director of the University's Technology Service Center from 1998 to 1999. From August 1999 to February 2000, he was a visiting faculty of the ECE Department, UCSB. Since August 2000, he has been the Chair of the EE Department of NTHU. He also is the Director of the IC Design Technology Center of the University. Dr. Wu was the Technical Program Chair of the IEEE Fifth Asian Test Symposium (ATS'96), and the General Chair of ATS'00. He is an Associate Editor for the Journal of the Chinese Institute of Electrical Engineers (JCIEE) and a Guest Editor of the JCIEE Speical Issue on Design and Test of System-on-Chip, and was a Guest Editor of the Journal of Information Science and Engineering (JISE), Special Issue on VLSI Testing. He received the Distinguished Teaching Award from NTHU in 1996, the Outstanding Electrical Engineering Professor Award from the Chinese Institute of Electrical Engineers (CIEE) in 1997, the Distinguished Research Award from National Science Council in 2001, the Industrial Collaboration Award from the Ministry of Education in 2001, and the Best Paper Award from the 2002 IEEE International Workshop on Design & Diagnostics of Electric Circuits & Systems. He is interested in design and testing of high performance VLSI circuits and systems. Dr. Wu is a life member of CIEE and a senior member of IEEE.