Copyright 1998 IEEE. Published in the Proceedings of VTS'98, April 1998 Monterey, California. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works, must be obtained from the IEEE. Contact: Manager, Copyrights and Permissions / IEEE Service Center / 445 Hoes Lane / P.O. Box 1331 / Piscataway, NJ 08855-1331, USA. Telephone: + Intl. 908-562-3966.

An Approach to Modeling and Testing Memories and its Application to CAMs*

Piotr R. Sidorowicz Janusz A. Brzozowski
Department of Computer Science
University of Waterloo
Waterloo, ON, Canada, N2L 3G1

Abstract

An approach to modeling and testing memories is presented and illustrated using an n-word by l-bit ($n \times l$) static content-addressable memory (CAM) array for cell input stuck-at faults. An input stuck-at fault model for a CAM is defined, and a test of length $7n + 2l + 5$ with 100% fault coverage with respect to this fault model is constructed. This test also detects all the usual cell stuck-at and transition faults. Finally, some design-for-testability (DFT) modifications facilitating a further reduction of this test's length are proposed.

1 Introduction

In this paper we describe a systematic approach to modeling and testing memories, and we illustrate it for CAM input stuck-at faults. The steps are: 1) transistor circuit analysis, 2) a low-level asynchronous model, 3) a high-level finite-state machine (FSM) model, and 4) a sequence of tests progressing from a single cell to the $(n \times l)$ CAM.

The basis for our approach to modeling and testing memories is the framework of [2]. This framework has been successfully utilized for modeling common faults in RAMs (See [3, 4, 5], for examples).

CAMs are word-oriented storage devices that allow instantaneous searches of the memory content for a given key word. This type of memory plays a vital role in caches, table-lookaside buffers, ATM switches, and other ASICs where searching is a critical operation. We focus on a CAM based on the cell shown in Figure 1(a), as this design is commonly used in the telecommunications industry [1, 6, 7].

An *input stuck-at fault model* is defined for the CAM. This fault model consists of any stuck-at fault which affects input lines of the CAM cells.

The remainder of this report is organized as follows: A detailed model of the CAM cell's behavior is presented in Section 2. Section 3 describes the derivation of a more abstract formal model, suitable for high-level analysis. The fault model is presented in Section 4, where a test for a single cell CAM is given. In Sections 5, 6 and 7 the test derived for a single cell is extended to $(n \times 1)$, $(1 \times l)$ and $(n \times l)$ CAMs, respectively. Design-for-testability modifications facilitating further reductions of the test length without affecting fault coverage are proposed in Section 8. Section 9 contains some concluding remarks.

*This research was supported by the Natural Sciences and Engineering Research Council of Canada under grant No. OGP0000871.

2 Low-Level CAM Model

The circuit of the cell shown in Figure 1(a) consists of a *storage section* and a *comparison section*. The

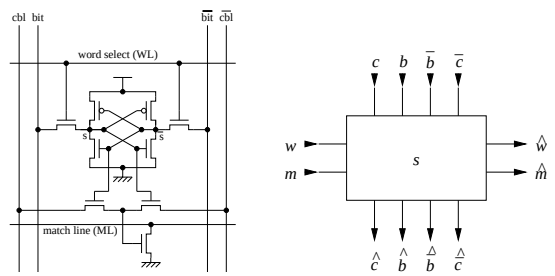

Figure 1: (a) Static CAM cell with dedicated compare lines, (b) its model.

storage section has two cross-coupled CMOS inverters, differential bit lines ($bit/\overline{bit}$) used for reading and writing data into a column of cells, and a word select line (WL) that enables these operations in a row of cells. In a quiescent state WL is driven low, the $bit/\overline{bit}$ lines are driven high and the cell stores either 0 or 1. During a 'write 1' or a 'write 0' operation, the $bit/\overline{bit}$ lines are driven to a true/complementary representation of the desired bit value. Raising and then lowering WL stores the bit in the cell. The 'write $\times$' operation preserves

the state of the cell. Its implementation requires both of the $bit/\overline{bit}$ lines to be driven high; proper margining of the transistors assures the preservation of the cell's state. A 'read' operation is performed by isolating both $bit/\overline{bit}$ lines, and then raising WL, thus causing one of the $bit/\overline{bit}$ lines to discharge. The voltage differential between the $bit/\overline{bit}$ lines is detected giving the content of the cell.

The comparison section consists of the bottom three transistors, the differential compare bit lines ($cbl/\overline{cbl}$) for matching operations, and the match line (ML). In a quiescent state $cbl/\overline{cbl}$ are driven low and ML is driven high. The 'compare' operation is similar to the 'read' operation. First, ML is isolated. Next, $cbl/\overline{cbl}$ are driven according to the desired search key. If a mismatch occurs, the bottom-most transistor will be forced to conduct, thus discharging ML. In the case of a 'compare $\times$' operation, both of $cbl/\overline{cbl}$ remain low, resulting in an unconditional match. Altogether, seven operations can be performed on a CAM cell: within the storage section - 'write 0', 'write 1', 'write $\times$' and 'read', and within the comparison section - 'compare 0', 'compare 1' and 'compare $\times$'.

Since no clock signal is supplied to individual cells, a single CAM cell can be viewed as an asynchronous sequential circuit, whose behavior can be modeled by the finite state machine of Figure 1(b). Since $bit/\overline{bit}$ lines and ML are used for both input and output in the circuit, they are represented by separate variables in the model. For brevity we use b, $\bar{b}$, w, etc. for bit, $\overline{bit}$, WL, etc. Now, b, $\bar{b}$ and m are inputs, and $\widehat{b}$, $\widehat{\bar{b}}$, and $\widehat{m}$ are outputs. Although WL and $cbl/\overline{cbl}$ are not usually meant to provide any output, monitoring the state of these lines, if possible, might improve testability. So, we generalize our model to include these lines; w, c and $\bar{c}$ for input, and $\widehat{w}$, $\widehat{c}$ and $\widehat{\bar{c}}$ for output.

The total state of the cell is defined by the values on the input and output lines of the cell, and by its internal state s. It is represented by the 13-tuple: $C_T = (w, b, \bar{b}, c, \bar{c}, m, s, \widehat{w}, \widehat{b}, \widehat{\bar{b}}, \widehat{c}, \widehat{\bar{c}}, \widehat{m})$. *However, it turns out that in the correct cell, as well as in the presence of input stuck-at faults, the output values are identical to those of the inputs; hence, we omit them.* The "reduced" total state is symbolically represented by $C = w\ b\bar{b}\ c\bar{c}\ m \cdot s$, where the input variables have been separated by spaces for readability, and the $\cdot$ has been inserted to separate the input variables from the internal state.

The domain of each variable in C is the set $Y = \{0, 1, \tilde{0}, \tilde{1}, I\}$. The values 0 and 1 represent lines driven to the logic values 0 and 1 respectively, while $\tilde{0}$ and $\tilde{1}$ denote lines that were first discharged and then isolated (*floated low*) or precharged and then isolated (*floated high*). Symbol I stands for an intermediate logic value, which is caused in a CMOS circuit when both pull-up and pull-down transistors conduct.

The CAM cell has two possible initial states: $C =$ 0 11 00 $1 \cdot 0$ and $C =$ 0 11 00 $1 \cdot 1$.

Table 1: Write operations in a fault-free CAM cell.

Write operations (s = 1)			
	W-0	W-1	W-$\times$
Description	$w\ b\bar{b}\ c\bar{c}\ m \cdot s$	$w\ b\bar{b}\ c\bar{c}\ m \cdot s$	$w\ b\bar{b}\ c\bar{c}\ m \cdot s$
initial state	0 11 00 1·1	0 11 00 1·1	0 11 00 1·1
set $bit/\overline{bit}$	0 01 00 1·1	0 10 00 1·1	0 11 00 1·1
raise WL	1 01 00 1·1	1 10 00 1·1	1 11 00 1·1
new state	1 01 00 1·0	1 10 00 1·1	1 11 00 1·1
lower WL	0 01 00 1·0	0 10 00 1·1	0 11 00 1·1
raise $bit/\overline{bit}$	0 11 00 1·0	0 11 00 1·1	0 11 00 1·1

The behavior of a cell is represented by sequences of events that take place during the seven operations. Table 1 lists events that occur during 'write' operations to a cell whose state s is 1. By events we mean changes in the value of the total state C. The occurrences of these events are ordered from top to bottom. Note that the top and bottom entries in each column are initial states.

Example: 'Write 0' operation. Initial state of the cell: 0 11 00 $1 \cdot 1$. First, b is lowered: 0 01 00 $1 \cdot 1$. Then w is raised: 1 01 00 $1 \cdot 1$. As a result, s changes: 1 01 00 $0 \cdot 0$. Next, w is lowered: 0 01 00 $0 \cdot 0$. Finally, both b and $\bar{b}$ are raised: 0 11 00 $0 \cdot 0$.

An exhaustive analysis of all operations is given in [8]. These analyses are done under the assumption that operations occur one at a time. Modeling concurrent operations is an open research topic.

At this level of detail, where any input can be controlled and any output observed, testing is a trivial process. It would suffice to compare the state of each line with its expected value; any disagreement would signify the presence of a fault. Unfortunately, this detail of monitoring is not feasible in any real circuit; thus, a more abstract model is necessary.

3 High-Level CAM Model

Most testing algorithms utilize sequences of 'read', 'write' and 'compare' operations as input, and observe the resulting output. Accordingly, we introduce a high-level model of a CAM cell, derived from the low-level model of Section 2.

We represent the behavior of a fault-free CAM cell as a finite-state machine (FSM), which is shown in

Figure 2(a). The input x of this FSM comprises all

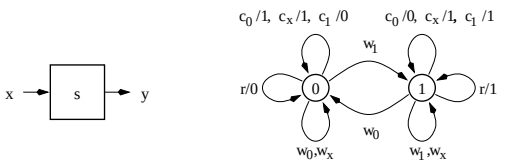

Figure 2: (a) Simplified behavioral model (b) Behavior of a fault-free cell.

seven operations of a CAM cell, and the output y comprises the responses to these operations without regard to the output's origin, i.e., the *bit*/$\overline{bit}$ lines or *ML*. State s represents the value stored by the cell. Formally, a *fault-free CAM cell* is a Mealy automaton $M = (Q, X, Y, \delta, \lambda)$, where $Q = \{0, 1\}$ is the set of states, $X = \{r, w_0, w_1, w_\times, c_0, c_1, c_\times\}$ is the set of input symbols, $Y = \{0, 1, I, \$\}$ is the set of output symbols, where \$ is a formal symbol denoting lack of output during 'write' operations, and the next-state function δ and the output function λ are defined by

$$\delta(q, x) = \begin{cases} 0, & \text{if } x = w_0, \\ 1, & \text{if } x = w_1, \\ q, & \text{otherwise} \end{cases}$$

and

$$\lambda(q, x) = \begin{cases} q, & \text{if } x = r, \\ 1, & \text{if } x = c_p \text{ and } q = p, \\ 0, & \text{if } x = c_p \text{ and } q \neq p, \\ 1, & \text{if } x = c_\times, \\ \$, & \text{otherwise.} \end{cases}$$

This automaton is depicted in Figure 2(b), where the symbol \$ has been omitted for clarity.

Example: For 'write 1' in state 0, $\delta(0, w_1) = 1$ and $\lambda(0, w_1) = \$$. For 'compare 1' in state 0, $\delta(0, c_1) = 0$ and $\lambda(0, c_1) = 0$.

4 Fault Model

A low-level behavioral analysis has been performed for every potential input stuck-at fault under the single-fault assumption [8]. Subsequently, high-level models for each of these faults have been developed.

Example: 'Write 0' operation in the presence of the $\bar{b}$-*sa-0* fault. Initial state: 0 10 00 $1 \cdot 1$. First, b is lowered: 0 00 00 $1 \cdot 1$. Then w is raised: 1 00 00 $1 \cdot 1$. Since both b and $\bar{b}$ are low, s becomes indeterminate: 1 00 00 $1 \cdot I$. Next, w is lowered: 0 00 00 $1 \cdot I$. Finally, both b and $\bar{b}$ are raised: 0 10 00 $1 \cdot 0|1$, and eventually s becomes 0 or 1.

From the low-level model we construct an automaton. In this example, only operations w_0, $w_\times$ and r are incorrect.

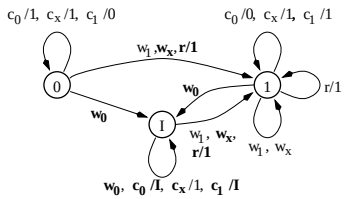

Figure 3: Faulty behavior of a CAM cell: $\bar{b}$-*sa-0*.

As the example indicates, the $\bar{b}$-*sa-0* and *b-sa-0* faults increase the state set Q by the "indeterminate" state I. State I can be interpreted in two ways. From an "asynchronous" viewpoint it represents a temporary, metastable state, where the cell holds some indeterminate logic value. From the "synchronous" perspective it represents the loss of information regarding the current state due to the non-deterministic resolution of a metastable state. In either case, state I is not considered as an initial state.

The comparison of each of the faulty machines to the fault-free CAM cell has led to the derivation of simple tests for each fault. Table 2 lists some of the shortest tests ending in a 'compare' operation. These tests have been generated by the OBSERVER[1] program for all input stuck-at faults except the *w-sa-1* fault. We refer to the faults that are detectable in a single cell as *independently testable*. The use of $w_\times$ and 'compare' operations rather than 'read' operations for the purpose of propagating faulty responses is dictated by the unreliable output of the 'read' operation for four faults related to the storage section of the cell: *b-sa-1*, $\bar{b}$-*sa-1*, *w-sa-0* and *w-sa-1*. In the last example a $w_0 w_\times$ forces the faulty cell to state 1, whereas a good cell would be in state 0. A w_0 alone is not enough to force the faulty cell to the erroneous state. See [8] for details.

The *w-sa-1* fault automaton is identical to that of the fault-free CAM cell; therefore, no test exists for a single cell. This is the only input stuck-at fault that is not independently testable. However, it is detectable in conjunction with other words, as will be seen later.

[1] OBSERVER is a program for diagnosing and testing sequential machines. It is based on the theory developed in [2] and was written at the University of Waterloo by C.-J. Shi; additional features were added by P. Kwiatkowski and P. R. Sidorowicz.

Table 2: Summary of input stuck-at faults.

Fault	Test Sequence	Faulty Response	Fault-Free Response
b-sa-0	$\mathbf{w_1 w_\times c_1}$	0	1
b-sa-1	$\mathbf{w_1 w_0 c_0}$	0	1
$\bar{b}$-sa-0	$\mathbf{w_0 w_\times c_0}$	0	1
$\bar{b}$-sa-1	$\mathbf{w_0 w_1 c_1}$	0	1
w-sa-0	$\mathbf{w_1 c_1 w_0 c_0}$	1 0 or 0 1	1 1
w-sa-1	see Section 5		
c-sa-0	$\mathbf{w_0 c_1}$	1	0
c-sa-1	$\mathbf{w_0 c_0}$	0	1
$\bar{c}$-sa-0	$\mathbf{w_1 c_0}$	1	0
$\bar{c}$-sa-1	$\mathbf{w_1 c_1}$	0	1
m-sa-0	$\mathbf{c_1 c_0}$	0 0	1 0 or 0 1
m-sa-1	$\mathbf{c_1 c_0}$	1 1	1 0 or 0 1

Partly with the aid of the OBSERVER program, we have found a test

$$T_{isaf} = w_0 w_1 w_\times c_1 c_0 w_0 w_\times c_0 c_1$$

of length 9 that detects all independently testable input stuck-at faults. This test is *irredundant*, since the removal of any of the input symbols in the test will result in some fault becoming undetectable. Tests for individual faults that are included in T_{isaf} are indicated in Table 2. With the exception of c-sa-0 and $\bar{c}$-sa-0 faults, these tests have been chosen because their response in a fault-free cell is a match, and in a faulty cell, a mismatch. Thus, the CAM's property, that a match line discharges in the presence of even a single bit mismatch, can be efficiently utilized.

As indicated in [8], a cell stuck-at-0 (stuck-at-1) fault is similar to a cell with w-sa-0, when that cell is started in state 0 (state 1). This fault is detected by $w_1 c_1$ ($w_0 c_0$). Also, a cell unable to undergo the $1 \rightarrow 0$ ($0 \rightarrow 1$) transition is similar to b-sa-1 ($\bar{b}$-sa-1). This fault is detected by $w_1 w_0 c_0$ ($w_0 w_1 c_1$).

5 Extension to $(n \times 1)$ CAM

We now consider an n-bit CAM where each word consists of a single cell. At this point, some assumptions are made about the peripheral circuitry in a fault-free CAM: 1) The address decoder can raise at most a single write line, 2) Match detection is done through a *hit* line, a *multi-hit* line, and an encoder that returns the address k of the highest priority match line[2].

[2]By convention, address 1 has the highest, and address n the lowest priority.

The model of the correct $(n \times 1)$ CAM as an FSM is given in [8]. The inputs $r^i, w_0^i, w_1^i, w_\times^i$ denote the 'read', 'write 0', 'write 1' and 'write $\times$' operations on word i, respectively, and $c_0, c_1, c_\times$ denote 'compare' operations with the contents of every cell in the CAM.

As indicated earlier, w-sa-1 is not independently testable and affects all the cells along the faulty *WL*. However, this fault is detectable in the following manner: Assume w^i-sa-1. Since word i is always accessed, a 'read' or 'write' may be applied to two words simultaneously. Note that if the two accessed words contain opposing values, 'read' operations will yield unreliable results. A possible test is:

$$T_{w\text{-}sa\text{-}1} = (w_0^i)^\updownarrow w_1^n c_1 w_0^n w_1^u c_1,$$

where $1 \leq i \leq n$ and $1 \leq u < n$. The symbol $(\ )^\updownarrow$ denotes n operations done in order either from n to 1 or from 1 to n; hence the $\updownarrow$. Initially, 0s are written into every word. If word u is the faulty word, $u \neq n$, then w_1^n will write 1 to both u and n. The first c_1 will generate a multiple hit and the returned value of k will indicate the address of the faulty word, as the faulty word has a higher priority than word n. If word n is the faulty word, then word u is not, so w_0^n restores the initial state of the CAM, w_1^u will write 1 to words u and n, and the second c_1 will generate a multiple hit. Here the value of k is ignored, because the location of the faulty word is known to be n. Thus, the occurrence of a multiple hit indicates the w-sa-1 fault.

The test T_{isaf} is modified into a march test

$$T_{n\text{-}isaf} = (w_0^i)^\updownarrow (w_1^i w_\times^i c_1)^{n\downarrow 1} c_0 (w_0^i w_\times^i c_0)^{n\downarrow 1} c_1$$

for the $(n \times 1)$ CAM. The symbol $(\ )^{n\downarrow 1}$ denotes the direction of the march elements: from n to 1. This direction is dictated by the priority scheme used in the match line encoder and always follows from the lowest to the highest priority. First, all the words are initialized to 0. Then, in the fault-free CAM, for each $w_1^i w_\times^i$ in a march element, the subsequent c_1 input should produce a value of k: $k = i$. Multiple hits during this test are expected; thus the status of the *multi-hit* line must be ignored. The mismatching c_0 is performed once per march element, on a CAM uniformly filled with 1s, and should produce a value $k = 0$, indicating a global mismatch. Any hit, i.e., any $k > 0$, indicates a fault. The rest of the test sequence is similar, with 0 and 1 interchanged. 100% fault coverage under the input stuck-at fault model is achieved by combining $T_{n\text{-}isaf}$ with $T_{w\text{-}sa\text{-}1}$:

$$\begin{aligned} T_{n\text{-}saf} \quad = \quad & (w_0^i)^\updownarrow (w_1^i w_\times^i c_1)^{n\downarrow 1} c_0 \\ & (w_0^i w_\times^i c_0)^{n\downarrow 1} c_1 (w_1^n c_1 w_0^n w_1^u c_1). \end{aligned}$$

We remind the reader that we are using the single-fault assumption. Note that the initial $(w_0^i)^{\updownarrow}$ of $T_{w\text{-}sa\text{-}1}$ has been dropped, as the CAM is expected to hold only 0s at the end of $T_{n\text{-}isaf}$. Note also that $w_\times$ is not needed in the last part of the test, because the faults $b\text{-}sa\text{-}0$ and $\bar{b}\text{-}sa\text{-}0$ which require $w_\times$ would have been detected by the first part of the test.

The $T_{n\text{-}saf}$ test has $5n+3$ 'write' and $2n+4$ 'compare' operations, and has an overall length of $7n+7$.

6 Extension to $(1 \times l)$ CAM

We now consider a single row of cells comprising a CAM word. The model of the correct behavior of this CAM is shown in [8]. Recall, that a match line is a 'wired AND' of match responses of all cells in a word.

Faults affecting the $bit/\overline{bit}$ lines of any of the l cells are detectable by performing the respective tests on all cells in parallel since these faults manifest themselves with a mismatch. Detection of the $w\text{-}sa\text{-}0$, $m\text{-}sa\text{-}0$ and the $m\text{-}sa\text{-}1$ faults is also done in parallel, as these faults affect the entire word in the same manner. Detection of each of the $c^j\text{-}sa\text{-}0$ and $\bar{c}^j\text{-}sa\text{-}0$ faults, where $1 \le j \le l$, is more complex. Their respective tests have a mismatch as a fault-free response and a match as a faulty one; thus, they cannot be applied to all the cells in parallel, as no faulty response would ever be propagated along the ML. To propagate a faulty response along the ML, a mismatch must be attempted on each cell j individually, while simultaneously applying a $c_\times$ to the remaining $l-1$ cells. Let $\times^{j-1}0\times^{l-j}$ be the l-bit word with 0 in position j and $\times$ in every other position. The word $\times^{j-1}1\times^{l-j}$ is similarly defined. The symbol $[c_{\times^{j-1}0\times^{l-j}}]^{\leftrightarrow}$ represents a sequence of l 'compare' operations to words of the form $\times^{j-1}0\times^{l-j}$, where j varies from 1 to l or from l to 1. The extension of the T_{isaf} test to the $(1 \times l)$ CAM takes the form

$$\begin{aligned} T_{l\text{-}isaf} \;=\; & w_{0\ldots0}w_{1\ldots1}w_{\times\ldots\times}c_{1\ldots1}[c_{\times^{j-1}0\times^{l-j}}]^{\leftrightarrow} \\ & w_{0\ldots0}w_{\times\ldots\times}c_{0\ldots0}[c_{\times^{j-1}1\times^{l-j}}]^{\leftrightarrow}, \end{aligned}$$

where $w_{0\ldots0}$ denotes the writing of the all-0 word, etc.

The $T_{l\text{-}isaf}$ test has 5 'write' and $2l+2$ 'compare' operations and has an overall length of $2l+7$.

7 Extension to $(n \times l)$ CAM

The combination of both extensions yields the specification for the behavior of a $(n \times l)$ CAM [8]. The following test provides 100% fault coverage under the input stuck-at fault model:

$$\begin{aligned} T_{saf} \;=\; & (w_{0\ldots0}^i)^{\updownarrow}(w_{1\ldots1}^i w_{\times\ldots\times}^i c_{1\ldots1})^{n\downarrow 1}[c_{\times^{j-1}0\times^{l-j}}]^{\leftrightarrow} \\ & (w_{0\ldots0}^i w_{\times\ldots\times}^i c_{0\ldots0})^{n\downarrow 1}[c_{\times^{j-1}1\times^{l-j}}]^{\leftrightarrow} \\ & (w_{1\ldots1}^n c_{1\ldots1} w_{0\ldots0}^n w_{1\ldots1}^u c_{1\ldots1}). \end{aligned}$$

This test consists of two parts, analogous to those of the $T_{n\text{-}saf}$ test. The first part consists of an initialization which sets all words to $0\ldots0$, followed by two march elements. For a fault-free CAM, each of the march elements should result in n hits with $k=i$. Each march element should be followed by l mismatches at $k=0$. Any other response indicates a fault. In the second part (last line), multiple hits are monitored, since they indicate faults.

The T_{saf} test has $5n+3$ 'write' and $2n+2l+2$ 'compare' operations. Thus, the length of our test for all input stuck-at faults in an $(n \times l)$ CAM is $7n+2l+5$.

8 DFT Suggestions

The length of our test for input stuck-at faults is linear in the number of bits in the CAM. This test can be substantially shortened by certain DFT enhancements.

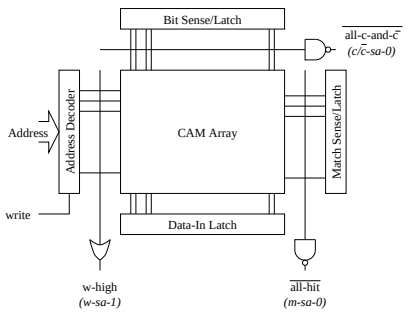

Figure 4: DFT suggestions.

The enhancements of Figure 4 target faults that contribute most to the length of the test:

- An $\overline{all\text{-}hit}$ output, indicating that a match has not been detected on every word simultaneously. Note that this output directly detects the $m\text{-}sa\text{-}0$ fault. Given this additional output, test T_{saf} can be modified to

$$\begin{aligned} T'_{saf} \;=\; & (w_{0\ldots0}^i)^{\updownarrow}(w_{1\ldots1}^i w_{\times\ldots\times}^i)^{n\downarrow 1}c_{1\ldots1}[c_{\times^{j-1}0\times^{l-j}}]^{\leftrightarrow} \\ & (w_{0\ldots0}^i w_{\times\ldots\times}^i)^{n\downarrow 1}c_{0\ldots0}[c_{\times^{j-1}1\times^{l-j}}]^{\leftrightarrow} \\ & (w_{1\ldots1}^n c_{1\ldots1} w_{0\ldots0}^n w_{1\ldots1}^u c_{1\ldots1}), \end{aligned}$$

  the length of which is reduced to $5n+2l+7$.

- An $\overline{all\text{-}c\text{-}and\text{-}\bar{c}}$ output, indicating that all c and all $\bar{c}$ inputs are high. This output would eliminate

elaborate testing for the *c-sa-0* and $\bar{c}$*-sa-0* faults. As a result, T'_{saf} can be further reduced to

$$\begin{aligned} T''_{saf} \;=\; & (w^i_{0\ldots0})^{\updownarrow}(w^i_{1\ldots1}w^i_{\times\ldots\times})^{n\downarrow1}c_{1\ldots1} \\ & (w^i_{0\ldots0}w^i_{\times\ldots\times})^{n\downarrow1}c_{0\ldots0}c_{1\ldots1} \\ & (w^n_{1\ldots1}c_{1\ldots1}w^n_{0\ldots0}w^u_{1\ldots1}c_{1\ldots1}), \end{aligned}$$

which has length $5n + 8$.

- A *w-high* output, indicating that at least one write line is active. This output can directly detect any *w-sa-1* fault, as well as *w-sa-0* faults during 'write' operations; hence, the suffix of the test T''_{saf} , $(w^n_{1\ldots1}c_{1\ldots1}w^n_{0\ldots0}w^u_{1\ldots1}c_{1\ldots1})$, can be omitted. If, however, the *w-high* output is unavailable, the *w-sa-1* fault can still be detected by reducing the above-mentioned suffix to $([\bar{b}^l = 0\ldots0]c_{1\ldots1})$, where $[\bar{b}^l = 0\ldots0]$ represents setting all the $\overline{bit}$ lines to ground — like the first step in the 'write 0' operation — without actually raising any of the *WL*s. This will result in the word affected by the *w-sa-1* fault having $1\ldots1$ written into it. A match resulting from a subsequent $c_{1\ldots1}$ operation indicates the presence of this fault.

- The ability to perform a *bulk-write* operation to all words simultaneously, in conjunction with the $\overline{all\text{-}hit}$ output, will result in a test length independent of the number of words. For example, the length of T'_{saf} would be reduced to $2l + 12$, and the length of T''_{saf} to 13.

Although these enhancements operate in *test mode*, some of them may adversely affect the *mission mode* operation of the CAM; therefore, a compromise between testability and performance may be necessary.

9 Conclusions

Using a static CMOS CAM as an example, we have developed a fault modeling and test derivation methodology. The modeling steps include: circuit analysis, asynchronous behavior analysis, and a finite automaton representation of the fault-free and faulty circuits. An input stuck-at fault model has been defined for an n-word by l-bit static CMOS CAM. A test with 100% fault coverage with respect to this fault model has been constructed. This test also detects all the usual cell stuck-at and transition faults. It has also been demonstrated that 'read' operations are inadequate sources of output for testing static CMOS CAMs, and that the 'compare' operations are a more reliable choice. In addition, some DFT enhancements that reduce the test length have been suggested.

The effect of other types of faults on the operation of this CAM cell is the topic of continuing research.

Acknowledgments

We thank Ken Schultz of Nortel Corporation's Memory Development Team, for providing extensive information regarding various defects and their effects on the behavior of a CAM. We also express our gratitude to Robert Berks, Tracey Bogue, Edward Dengler and all the members of the Maveric Group for their insightful comments and suggestions.

References

[1] H. Bergh, J. Eneland, and L-E. Lundström. A fault-tolerant associative memory with high-speed operation. *IEEE Journal of Solid-State Circuits*, 25(4):912–919, August 1990.

[2] J. A. Brzozowski and H. Jürgensen. A model for sequential machine testing and diagnosis. *Journal of Electronic Testing: Theory and Applications*, 3:219–234, 1992.

[3] B. F. Cockburn and J. A. Brzozowski. Near-optimal tests for classes of write-triggered coupling faults in RAMs. *Journal of Electronic Testing: Theory and Applications*, 3:251–264, 1992.

[4] A. J. van de Goor and B. Smit. The automatic generation of march tests. In *Records of the IEEE Workshop on Memory Technology, Design and Testing*, pages 86–91, San Jose, CA, August 1994. IEEE Computer Society Press.

[5] A. J. van de Goor and B. Smit. Automating the verification of memory tests. In *IEEE VLSI Test Symposium*, pages 312–318, Cherry Hill, NJ, April 1994. IEEE Computer Society Press.

[6] S. Kornachuk, L. McNaughton, R. Gibbins, and B. Nadeau-Dostie. A high speed embedded cache design with non-intrusive BIST. In *Records of the IEEE Workshop on Memory Technology, Design and Testing*, pages 40–45. IEEE, August 1994.

[7] K. J. Schultz. *CAM-Based Circuits for ATM Switching Networks.* PhD thesis, University of Toronto, 1996.

[8] P. R. Sidorowicz and J. A. Brzozowski. Testing for input stuck-at faults in content-addressable memories.
Technical Report CS-98-02, University of Waterloo, April 1998. ftp://cs-archive.uwaterloo.ca/cs-archive/CS-98-02/CS-98-02.ps.Z.