A Framework for Testing Special-Purpose Memories*

Piotr R. Sidorowicz
prsidoro@mavericsolutions.ca
Maveric Solutions
200 Clearview Avenue, suite 2328
Ottawa, ON, Canada, K1Z 8M2

Janusz A. Brzozowski
brzozo@uwaterloo.ca
Department of Computer Science
University of Waterloo
Waterloo, ON, Canada, N2L 3G1

Abstract

Current memory testing methods rely on fault models that are inadequate to accurately represent potential defects that occur in modern, often specialized, memories. To remedy this we present a formal framework for modeling and testing special-purpose memories. Our approach uses three models: the transistor circuit, the event-sequence model and finite-state machines. The methodology is explained using the example of a content-addressable memory (CAM). The fault model we describe comprises input stuck-at, transistor and bridging faults. We show that functional tests can reliably detect all input stuck-at faults, most transistor faults (including all stuck-open faults), and about 50% of bridging faults. The remaining faults are detectable by parametric tests. A test of length $7n + 2l + 9$ that detects all the reliably testable faults in an n-word by l-bit CAM is presented. A CAM test by Giles & Hunter is evaluated with respect to the input stuck-at faults. It is shown that this test fails to detect certain faults; it can be modified to achieve full coverage at the cost of increased length.

1 Introduction

Memory circuits constitute a large part of integrated circuit production. High bit densities of modern memory chips render them particularly prone to failure, thus magnifying the need for thorough testing methods that are efficient enough to be economically justifiable.

*This research was supported by the Natural Sciences and Engineering Research Council of Canada under grant No. OGP0000871. This work was done while the first author was at the University of Waterloo. Parts of this work have been previously presented in [25], [27], [29].

As technologies and cell designs change and as the circuits become smaller, it is no longer clear whether fault models designed some ten years ago are still applicable to the memories of today. This is particularly true for special-purpose memories that incorporate additional, non-standard circuitry. Although new tests for these memories are being developed [9], the approach taken is still rather *ad-hoc*, and results in a number of highly abstract fault models, detached from the design and the technology for which they were intended.

In this paper we establish simple formal fault models for specific memory cell designs. As a representative of a special-purpose memory we choose a CAM, a word-oriented storage device that is being utilized increasingly often in digital designs for its parallel search capabilities. CAMs are usually found embedded in various high performance application-specific integrated circuits (ASICs).

We introduce our framework by applying it to a transistor circuit of a static CMOS CAM cell utilized in telecommunication ASICs [13]. We consider faults at the transistor circuit level, namely line or node stuck-at faults, transistor stuck-(on/open) faults and (resistive/hard) bridging faults. For each of these faults we derive a finite-state machine (FSM) model, and then use the FSM model to derive tests. We show that only a fraction of the faults that can occur in the CAM cell correspond to the well-established fault types such as *cell stuck-at* faults. This shows a benefit of our approach.

Our fault model provides a formal foundation for determining shortest test sequences for various faults in the model. It also provides a basis for comparisons among methods currently used for testing CAMs in terms of coverage of faults that are specific to a particular CAM cell circuit design.

The remainder of this paper is organized as follows: Section 2 discusses fault modeling for memories. An outline of our framework is given in Section 3. Content-addressable memories are briefly discussed in Section 4. In Section 5 we develop our model for a fault-free CAM. The models of faulty CAMs are the topic of Section 6. Fault detectability results are summarized in Section 7. Test generation is described in Section 8. Section 9 illustrates the use of our methodology in test evaluation. Section 10 contains some concluding remarks.

2 Memory Fault Modeling

Detailed discussions of memory fault modeling techniques have been presented in the comprehensive studies of [8, 22]. Defects in memory circuits can be partitioned into two categories: global and local. Global defects are often related to the manufacturing process, and affect a large area of a wafer; they are readily detectable by parametric tests. Local or *spot* defects, such as dust particles on the chip or gate oxide pinholes, manifest themselves mostly as *functional* faults. They are interpreted at the layout level as broken wires, shorts between wires, missing contacts, extra contacts and newly created transistors.

The following high-level fault models have been often used in the past [8, 22]: cell stuck-at fault, cell stuck-open fault, multiple-cell access fault, data retention fault, coupling fault, transition fault (static memories only), and pattern-sensitivity fault (dynamic memories only).

With the development of special-purpose memories, such as multi-port-RAMs, CAMs and FLASH memories, the number of fault models is growing at an alarming rate [3, 8, 9, 15]. Many of these models fail to specify the design details and the underlying technology (Bipolar, NMOS, CMOS, BiCMOS) of the circuit for which they were designed. Consequently, tests designed to detect faults for a given fault model may not apply to functionally similar memories designed in another technology. To remedy this situation, we pay attention to application-specific aspects of the design. In particular, we choose to identify faults at the transistor level; this is a compromise between poor accuracy of a higher-level model and unmanageable complexity of lower-level models. We then systematically construct FSM models from the transistor models, and develop functional tests for the FSM fault model.

3 Outline of Methodology

In this section we present a brief outline of our formal framework for modeling and testing memories. Our framework can be split into three distinct processes: *fault model derivation*, *test generation* and *test evaluation*. These processes are outlined below:

Fault Model Derivation consists of three consecutive steps of increasing level of abstraction:

1. *Transistor Circuit Analysis.*
   In a fault-free circuit, the sequence of voltage changes is well known for each operation. After determining the defects to be modeled, we introduce simple faults into the cell's transistor circuit (e.g., input stuck-at faults, transistor stuck-(on/open) faults,

etc.). Changes in the behavior of the cell resulting from these faults are then recorded.

2. *Event-Sequence Model.*
   This is a low-level discrete model. Using the transistor circuit analysis, we describe the behavior of the cell during each operation in terms of logic states. Voltage changes are modeled as transitions between states; these transitions are referred to as *events*.

3. *Finite-State Machine Model.*
   We adopt a formalism developed by Brzozowski and Jürgensen [5] for sequential circuit testing and diagnosis. The FSM model is a higher level model. It describes the behavior of a cell in terms of cell operations. Each operation, consisting of several events, is treated as a transition in a finite-state machine. Metastability is handled by introducing state I which represents the loss of information about the current state of the cell.

Test Generation is done at the FSM level, and also consists of three consecutive steps:

1. Generation of *elementary* tests that detect individual faults in a cell. This is accomplished by a search for shortest distinguishing sequences of operations between the good and faulty FSMs.
2. Development of a test that detects all testable faults in a single cell. This test has to contain all elementary tests.
3. Extension of this test to an array of cells, accounting for design-specific functionality and *dependently testable*[1] faults.

Test Evaluation is a process of establishing fault coverage. It is, in essence, the test generation process in reverse:

1. Representation of the test from the perspective of an arbitrary single cell in the array.
2. Verification of the existence of elementary tests in the above representation.
3. Verification of detection of dependently testable faults.

[1]A fault is dependently testable when its detectability requires the use of another cell during testing. More details are given in Section 8.1.

4 Content-Addressable Memories

In a typical random-access memory (RAM) stored data is accessed by specifying the *address* of the location in memory where data resides. A content-addressable memory (CAM), on the other hand, can indicate the locations of data words that match a particular bit pattern, called a *search key*. This is accomplished by performing a simultaneous comparison of all memory locations with the search key.

Although general purpose CAM designs have been investigated [1], most CAMs manufactured today can be found in custom designs, often embedded in larger circuits. Many application-specific CAM configurations have been reported [2, 4, 6, 10, 11, 12, 13, 14, 18, 21, 24, 30]. The most significant functional differences among these configurations include the accessibility by address as well as content, match-and-update, resolution of multiple hits, synchronous or asynchronous operation, etc., resulting in diverse implementations of the CAM's peripheral circuitry. However, the design of the storage element of a core cell, in most cases [1, 2, 4, 6, 11, 13, 20, 21, 24] is similar and consists of a cross-coupled inverter circuit, such as those found in static RAMs (SRAMs). Different designs of the core cell's comparison circuitry represent attempts to address various electrical pitfalls such as data-dependent bit-line loads, or charge-sharing problems [23].

Notable exceptions to the static CAM schema are dynamic implementations of the core cell [17, 19, 31, 32]. They are more complex than those of a typical high-density dynamic RAM, due to the necessity of charge retention during the simultaneous data comparison process. Like all dynamic memory implementations, these CAMs have to incorporate additional refresh circuitry.

Although most of the CAMs manufactured today are designed for specific applications, they share the same general architecture. A generic architecture of a CAM is depicted in Figure 1. All CAM implementations consist of the following functional blocks:

- **data and mask register,** which stores a particular word according to which memory is to be accessed. The ability to mask irrelevant parts is usually provided.

- **response processing and word access unit,** which determines the functionality of the CAM. The implementation of this block is the main source of differences among various CAM designs. This block handles resolution of multiple hits and keeps track of unused locations. It often contains an address register for reading and writing in RAM mode.

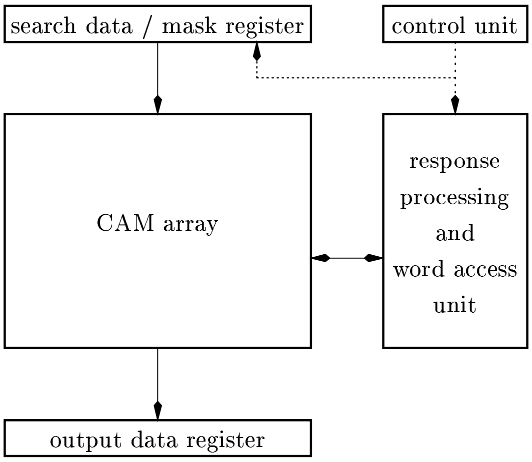


Figure 1: CAM: a block diagram.

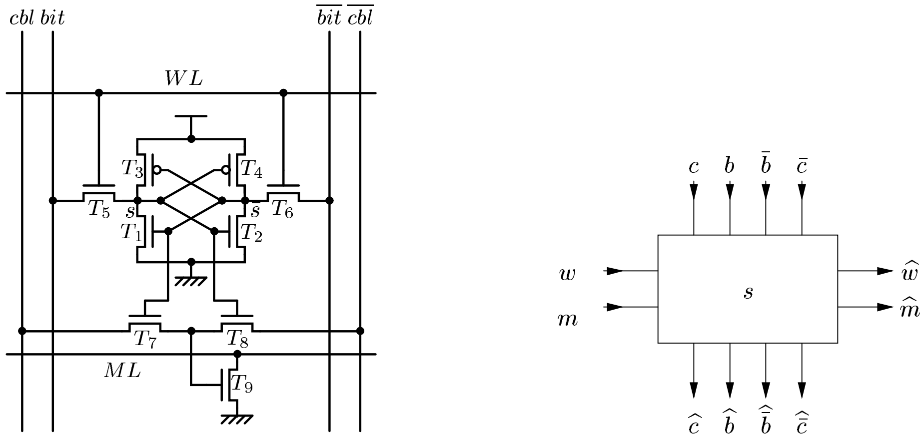
- **output data register,** which incorporates sensing circuitry that holds words picked up from bit lines during a read operation.
- **CAM array,** a homogeneous matrix of core cells which stores the data.
- **Control unit,** which coordinates the operation of the aforementioned blocks.

5 Fault-Free Models for a CAM Cell

5.1 Fault-Free Transistor Circuit

A common CMOS implementation of a static CAM cell [4, 13, 23, 24] is depicted in Figure 2 (a). It is composed of a typical six-transistor SRAM cell (*storage section*) and a three-transistor matching circuitry (*comparison section*). 'Read' and 'write' operations are identical to those in an SRAM cell [28]. Unlike SRAMs, CAMs have one more 'write' operation - 'write $\times$', which preserves the previous state of the cell. It is used for bit-masking purposes, where only a part of the word is to be overwritten. During a 'write $\times$' both *bit* and $\overline{bit}$ are driven to high voltage.

The process of searching is unique to the CAM, and is performed in the comparison section of the cell. In Figure 2 this section consists of transistors T_7, T_8 and T_9, a separate pair of differential lines $cbl/\overline{cbl}$ used for performing



(a) (b)

Figure 2: (a) A static CAM cell, (b) its model.

matching operations, and a match line *ML*. The *ML* line together with transistor T_9 constitute a wired-AND for all the cells in the memory word. This means that *ML* is isolated from ground as long as all the T_9 transistors are not conducting. Transistors T_7 and T_8 implement an XOR function between the *cbl* and $\overline{cbl}$ lines and values stored on the outputs of the inverters. Note that only one of these transistors conducts at any given time. Three operations can be performed: 'compare 0', 'compare 1' and 'compare $\times$' which results in an unconditional match.

All operations are performed by executing sequences of voltage changes on the cell's input lines. These sequences are determined by the memory's control circuitry.

5.2 Event-Sequence Model

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 block diagram of Fig. 2 (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 the CAM's testability. For this reason, we generalize our model to include these lines; w, c, $\bar{c}$ for input, and $\widehat{w}$, $\widehat{c}$, $\widehat{\bar{c}}$ for output.

The total state of the cell is defined by the values present 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 the considered faults, the output values are identical to those of the inputs; hence, we omit them for simplicity. 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 to show their functional separation and the symbol $\cdot$ has been inserted to separate the input variables from the internal state.

The domain of each variable in state C is the set $Y = \{0, 1, \tilde{0}, \tilde{1}\}$. 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 pre-charged and then isolated (*floated high*). The CAM cell has two possible initial total states: $C = 0\ 11\ 00\ 1 \cdot 0$ and $C = 0\ 11\ 00\ 1 \cdot 1$.

Example: 'Write 0' operation from state $s = 1$.

1. When the cell is in state 1, and no operations take place, the state of all the input lines is as follows: w, c and $\bar{c}$ are driven to ground, b, $\bar{b}$ and m are driven to V_{cc}. Node s is at V_{cc} by assumption.
2. The execution of 'write 0' begins by driving b to ground. This establishes a 0 on the differential $b/\bar{b}$ lines.
3. Next, w is driven to V_{cc}. This connects the $b/\bar{b}$ lines with $s/\bar{s}$ nodes.
4. As a result, s is driven (through b) to ground, and $\bar{s}$ (through $\bar{b}$) to V_{cc}. Thus transistors T_1 and T_4 conduct, and T_2 and T_3 do not conduct. At this time the cell changes state.
5. Line w is driven to ground; this disconnects $b/\bar{b}$ from $s/\bar{s}$.
6. Lastly both b and $\bar{b}$ are driven to V_{cc}.

We translate this process into the corresponding event sequence. By events we mean changes in the value of the total state C:

1. Initial state of the cell: .... $0\ 11\ 00\ 1 \cdot 1$
2. First, b is lowered: .... $0\ 01\ 00\ 1 \cdot 1$

3. Then w is raised:..1 01 00 1 · 1
4. As a result, the state s changes:....................1 01 00 1 · 0
5. Next, w is lowered:......................................0 01 00 1 · 0
6. Finally, both b and $\bar{b}$ are raised:....................0 11 00 1 · 0

We have performed a similar analysis for each CAM operation. The behavior of a cell is represented by sequences of events that take place during the seven operations as shown in Table 1. The occurrences of these events are ordered from top to bottom. Note that each operation starts in (top entry) and returns to (bottom entry) an initial state.

It should be noted that the analyses presented in Table 1 are done under the assumption that operations occur one at a time. However, 'compare' operations utilize a separate set of differential lines and thus some concurrent executions are feasible. For example, a 'compare' operation can be performed in parallel with a 'read' operation, but it cannot be performed until a 'write' operation is completed. Modeling concurrent operations is an open research topic.

Table 1: Read, write and compare operations in a fault-free CAM cell.

Read operations		
	$(s=0)$	$(s=1)$
Description	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 1·0	0 11 00 1·1
float $b/\bar{b}$	0 $\tilde{1}\tilde{1}$ 00 1·0	0 $\tilde{1}\tilde{1}$ 00 1·1
raise w	1 $\tilde{1}\tilde{1}$ 00 1·0	1 $\tilde{1}\tilde{1}$ 00 1·1
b or $\bar{b}$ discharges[2]	1 0$\tilde{1}$ 00 1·0	1 $\tilde{1}$0 00 1·1
read $b/\bar{b}$ & lower w	0 $\tilde{0}\tilde{1}$ 00 1·0	0 $\tilde{1}\tilde{0}$ 00 1·1
raise $b/\bar{b}$	0 11 00 1·0	0 11 00 1·1

Write operations						
	$(s=0)$			$(s=1)$		
	w_0	w_1	$w_\times$	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 $b\bar{b}$ $c\bar{c}$ $m{\cdot}s$	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 1·0	0 11 00 1·0	0 11 00 1·0	0 11 00 1·1	0 11 00 1·1	0 11 00 1·1
set $b/\bar{b}$	0 01 00 1·0	0 10 00 1·0	0 11 00 1·0	0 01 00 1·1	0 10 00 1·1	0 11 00 1·1
raise w	1 01 00 1·0	1 10 00 1·0	1 11 00 1·0	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 1·0	1 01 00 1·0	1 10 00 1·1	1 11 00 1·1
lower w	0 01 00 1·0	0 10 00 1·1	0 11 00 1·0	0 01 00 1·0	0 10 00 1·1	0 11 00 1·1
raise $b/\bar{b}$	0 11 00 1·0	0 11 00 1·1	0 11 00 1·0	0 11 00 1·0	0 11 00 1·1	0 11 00 1·1

Compare operations						
	$(s=0)$			$(s=1)$		
	c_0	c_1	$c_\times$	c_0	c_1	$c_\times$
Description	w $b\bar{b}$ $c\bar{c}$ $m{\cdot}s$	w $b\bar{b}$ $c\bar{c}$ $m{\cdot}s$	w $b\bar{b}$ $c\bar{c}$ $m{\cdot}s$	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 1·0	0 11 00 1·0	0 11 00 1·0	0 11 00 1·1	0 11 00 1·1	0 11 00 1·1
float m	0 11 00 $\tilde{1}$·0	0 11 00 $\tilde{1}$·0	0 11 00 $\tilde{1}$·0	0 11 00 $\tilde{1}$·1	0 11 00 $\tilde{1}$·1	0 11 00 $\tilde{1}$·1
set $c/\bar{c}$	0 11 01 $\tilde{1}$·0	0 11 10 $\tilde{1}$·0	0 11 00 $\tilde{1}$·0	0 11 01 $\tilde{1}$·1	0 11 10 $\tilde{1}$·1	0 11 00 $\tilde{1}$·1
m discharges	0 11 01 $\tilde{1}$·0	0 11 10 0·0	0 11 00 $\tilde{1}$·0	0 11 01 0·1	0 11 10 $\tilde{1}$·1	0 11 00 $\tilde{1}$·1
read m & ground $c/\bar{c}$	0 11 00 $\tilde{1}$·0	0 11 00 $\tilde{0}$·0	0 11 00 $\tilde{1}$·0	0 11 00 $\tilde{0}$·1	0 11 00 $\tilde{1}$·1	0 11 00 $\tilde{1}$·1
raise m	0 11 00 1·0	0 11 00 1·0	0 11 00 1·0	0 11 00 1·1	0 11 00 1·1	0 11 00 1·1

[2] There is a connection from b to V_{dd} when $s = 0$ and from $\bar{b}$ to V_{dd} when $s = 1$. But this connection is through a weak p-transistor and an n-transistor, and drivers for the $b/\bar{b}$ are not used. Hence, we still represent these cases as floating values.

5.3 FSM Model

Most testing algorithms utilize sequences of 'read', 'write' and 'compare' operations as input, and observe the resulting output; the behavior of a cell is often depicted as an FSM. Accordingly, we introduce an FSM model of a CAM cell.

Our FSM model is derived from the event-sequence model of Section 5.2. The example in the preceding section described the obvious: that a 'write 0' operation on a fault-free CAM cell in state 1 will result in a state change to 0. The significance of the event-sequence model, however, is the step-by-step description of how operations are implemented; this design-specific information is lost in the FSM model. This will become clearer when we consider faulty cells.

We represent the behavior of a fault-free CAM cell as a simplified block diagram, which is shown in Figure 3(a). The input x of this FSM comprises

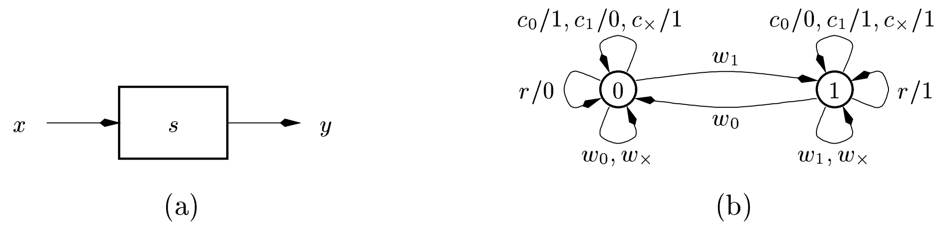


Figure 3: (a) Simplified behavioral model (b) Behavior of a fault-free cell.

all 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, $Y = \{0, 1, \$\}$ is the set of output symbols, where \$ is a formal symbol denoting lack of output during 'write' operations, and the transition 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 3(b), where the symbol \$ has been omitted for clarity.

Example: For 'write 0' in state 1, $\delta(1, w_0) = 0$ and $\lambda(1, w_0) = \$$. For 'compare 0' in state 1, $\delta(1, c_0) = 1$ and $\lambda(1, c_0) = 0$.

6 Faulty CAM

In this section we study CAM faults at the transistor level, and derive event-sequence and FSM models for these faults, paralleling the steps we used in the fault-free case.

In some faulty circuits the state of the cell may be neither 0 nor 1, but some intermediate voltage. Such a state may be metastable, meaning that this state will eventually resolve itself to either 0 or 1, though the exact outcome and the time of that outcome cannot be determined. To take care of this situation we introduce the indeterminate state I. State I can be interpreted in two ways. From an "asynchronous" point of view it represents a temporary state where the cell holds some indeterminate logic value. This interpretation is important due to the possibility of simultaneous operations in this type of CAM. From the "synchronous" perspective it represents the loss of information regarding the current state of the cell due to the non-deterministic resolution of a metastable state. We also use I as an output symbol representing an intermediate logic value, which is caused in a faulty CMOS circuit when both pull-up and pull-down transistors simultaneously conduct. The FSM model of the previous section in now modified as follows. The (faulty) set of states Q' is $Q \cup \{I\}$, and the (faulty) set of output symbols Y' is $Y \cup \{I\}$.

We have studied the following transistor-level fault types for CAMs: input stuck-at, state stuck-at, transistor stuck-(on/open), and bridging. Some of these faults resemble well known faults, such as cell stuck-at, transition and coupling faults. However, many of the faults analyzed have a distinct behavior and do not cleanly fit these categories. Several examples of transistor-level CAM faults and the corresponding fault models are given below.

6.1 Input Stuck-at Fault

Consider the operation 'write 0' applied to a CAM cell in state 1 in the presence of the $\bar{b}$-*sa-0* fault. We first analyze the faulty transistor circuit.

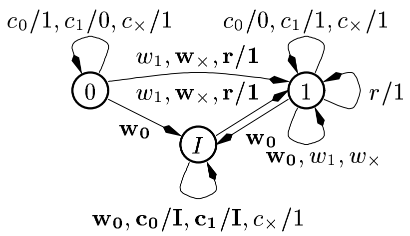
1. When the faulty cell is in state 1 and no operations take place, the state of all the input lines is as follows: w, c, $\bar{c}$ and $\bar{b}$ (because of the defect) are driven to ground, b, m and s are driven to V_{cc}.
2. The execution of 'write 0' begins by driving b to ground. Note that now both b and $\bar{b}$ are grounded.
3. Next w is driven to V_{cc}. This connects the $b/\bar{b}$ with $s/\bar{s}$ nodes.
4. Since both b and $\bar{b}$ are grounded, so are s and $\bar{s}$. The weak pull-up transistors, T_3 and T_4 conduct.
5. Now w is driven to ground. At this point the cell is metastable.
6. Lastly, b is driven to V_{cc}, but $\bar{b}$ remains grounded because of the defect.

We translate this transistor-level analysis into the corresponding event sequence model, which is shown in Figure 4 (a), along with the fault-free sequence. Erroneous values are indicated in boldface. Note that, although in the faulty cell s eventually becomes either 0 or 1, from a testing perspective the state remains unknown, and hence has the value I.

The event sequence tells us that the transition from state 1 in the presence of $\bar{b}$-*sa-0* fault is to an indeterminate state I. Thus in the FSM model, we introduce the transition from state 1 to state I under input w_0. We repeat this analysis for all the other operations. The complete FSM for this fault is given in Figure 4 (b).

Write 0 operation ($s = 1$)		
Description	Correct $w\ b\bar{b}\ c\bar{c}\ m{\cdot}s$	*b-sa-0* $w\ b\bar{b}\ c\bar{c}\ m{\cdot}s$
initial state	0 11 00 1·1	0 1**0** 00 1·1
set $b/\bar{b}$	0 01 00 1·1	0 0**0** 00 1·1
raise w	1 01 00 1·1	1 0**0** 00 1·1
new state	1 01 00 1·0	1 0**0** 00 1·**I**
lower w	0 01 00 1·0	0 0**0** 00 1·**I**
raise $b/\bar{b}$	0 11 00 1·0	0 1**0** 00 1·**I**

(a)



(b)

Figure 4: Faulty cell behavior ($\bar{b}$-*sa-0*): (a) w_0 when $s = 1$, (b) FSM.

6.2 Transistor Fault

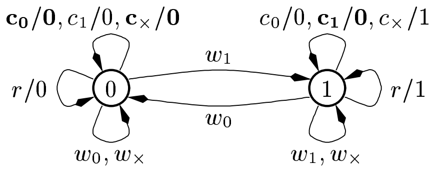
Consider the operation 'compare 1' applied to a CAM cell in state 1 in the presence of the T_7-*on* fault.

1. When the faulty cell is in state 1 and no operations take place, the state of all the input lines is as follows: w, c, $\bar{c}$ are driven to ground, and b, $\bar{b}$, m and s are driven to V_{cc}. Note that both the defective transistor T_7 and transistor T_8 conduct.
2. The execution of 'compare 1' begins by isolating m from V_{cc}. Line m carries a charge equal to V_{cc}.
3. Then, c is driven to V_{cc}; this establishes a 1 of the differential $c/\bar{c}$ lines.
4. The conducting transistors T_7 and T_8 act as a voltage divider. As a result, the voltage on the gate of T_9 rises high enough to discharge m.
5. The discharged m is read as a mismatch. At the same time both c and $\bar{c}$ are driven to ground.
6. Lastly, m is driven back to V_{cc}.

We translate this process into the event sequence shown in Figure 5 (a). This sequence shows that the transition from state 1 is back to state 1 and the output is $m = 0$. In the FSM model this sequence is depicted as a transition from state 1 to state 1 under input c_0 and output 0. All other operations have been analyzed in the same manner. The FSM for this fault is given in Figure 5 (b).

Compare 1 operation ($s = 1$)		
Description	Correct w $b\bar{b}$ $c\bar{c}$ $m{\cdot}s$	T_7-*on* w $b\bar{b}$ $c\bar{c}$ $m{\cdot}s$
initial state	0 11 00 1·1	0 11 00 1·1
float m	0 11 00 $\tilde{1}$·1	0 11 00 $\tilde{1}$·1
set $c/\bar{c}$	0 11 10 $\tilde{1}$·1	0 11 10 $\tilde{1}$·1
m discharges	0 11 10 $\tilde{1}$·1	0 11 10 **0**·1
read m & ground $c/\bar{c}$	0 11 00 $\tilde{1}$·1	0 11 00 **0**·1
raise m	0 11 00 1·1	0 11 00 1·1

(a)



(b)

Figure 5: Faulty cell behavior T_7-*on*: (a) c_1 when $s = 1$, (b) FSM.

6.3 Bridging Fault

Consider the operation 'compare 1' applied to a CAM cell in state 0 in the presence of the c-$\bar{c}$-*hrd* fault. This fault represents a non-resistive bridge (a short) between lines c and $\bar{c}$.

1. When the faulty cell is in state 0 and no operations take place, the state of all the input lines is as follows: w, c, $\bar{c}$ are driven to ground and b, $\bar{b}$ and m are driven to V_{cc}. Node s is driven to ground.

2. The execution of 'compare 1' begins by isolating m from V_{cc}. Line m carries a charge equal to V_{cc}.

3. Then c is driven to V_{cc}; however, due to the short between c and $\bar{c}$, c remains grounded[3].

4. Since both c and $\bar{c}$ remain grounded, T_9 does not conduct, and hence, m does not discharge.

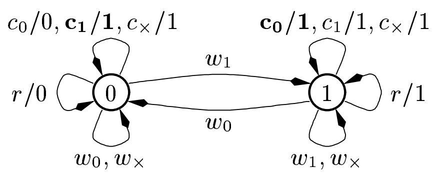
5. Next, m is read as a match. At the same time and c and $\bar{c}$ are driven to ground.

6. Finally, m is is driven back to V_{cc}.

We translate this process into the event sequence shown in Figure 6 (a). This sequence shows that the transition from state 0 is back to state 0 and the output is $m = 1$. In the FSM model this sequence is depicted as a transition from state 0 to state 0 under input c_1 and output 1. All other operations have been analyzed in the same manner. The FSM model for this fault is given in Figure 6 (b).

Compare 1 operation ($s = 0$)		
	Correct	c-$\bar{c}$-hrd
Description	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 $1 \cdot 0$	0 11 00 $1 \cdot 0$
float m	0 11 00 $\tilde{1} \cdot 0$	0 11 00 $\tilde{1} \cdot 0$
set $c/\bar{c}$	0 11 10 $\tilde{1} \cdot 0$	0 11 **0**0 $\tilde{1} \cdot 0$
m discharges	0 11 10 $0 \cdot 0$	0 11 **00** $\tilde{\mathbf{1}} \cdot 0$
read m & ground $c/\bar{c}$	0 11 00 $0 \cdot 0$	0 11 00 $\tilde{\mathbf{1}} \cdot 0$
raise m	0 11 00 $1 \cdot 0$	0 11 00 $1 \cdot 0$

(a)



(b)

Figure 6: Faulty cell behavior c-$\bar{c}$-hrd: (a) c_1 when $s = 0$, (b) FSM.

The process of obtaining FSMs from the transistor circuit analysis is an arduous one. This process, however, needs to be done only once for a given cell design. The result is an accurate representation of faulty cell behaviors in a well-known FSM format. From now on we use only the FSM model.

[3]We assume that pull-down circuitry dominates pull-up circuitry.

7 Fault Detectability Results

In this paper we consider only single faults. We have found FSMs for the following fault types: input line stuck-at, state stuck-at, transistor stuck-(on/open), and bridging. Bridging faults include resistive and non-resistive shorts between input lines or nodes within a cell and between input lines of adjacent cells. These faults constitute our fault model. We have categorized these faults with respect to their detectability by functional tests:

- **Independently testable faults**
  where the faulty FSM is distinguishable from the fault-free FSM. For all such faults we have found short tests, which we call elementary. Elementary tests for the independently testable faults are listed in Table 2.

- **Dependently testable faults**
  where the faulty FSM is identical to the fault-free FSM, and hence no test exists; however, the nature of the fault makes the cell vulnerable to change when a different cell is accessed (e.g., *w-sa-1* fault). Details are given in Section 8.1.

- **Faults that are not reliably testable**
  where the faulty FSM differs from the fault-free FSM, but the distinguishing output is I (e.g., T_3-*on* fault, as shown in Figure 7), or where the faulty FSM is identical to the fault-free FSM (e.g., *b-w-res* fault). These faults require parametric tests (see Table 3).

Table 2: Summary of elementary tests for a CAM.

Elementary Test	Faults Detected
$w_1 w_\times c_1$	*b-sa-0*, T_1-*on*, T_2-*open*, T_3-*open*
$w_1 w_0 c_0$	*b-sa-1*, T_6-*open*, *b-w-hrd*
$w_0 w_\times c_0$	$\bar{b}$-*sa-0*, T_1-*open*, T_2-*on*, T_4-*open*
$w_0 w_1 c_1$	$\bar{b}$-*sa-1*, T_5-*open*, $\bar{b}$-*w-hrd*, $\bar{c}$-*m-res*
$w_1 c_1 w_0 c_0$	*w-sa-0*
$w_0 c_1$	*c-sa-0*, T_7-*open*, *b-m-hrd* $\bar{b}$-*m-hrd*, *c-*$\bar{c}$*-hrd*
$w_0 c_0$	*c-sa-1*, *s-sa-1*, $\bar{s}$-*sa-0*, T_5-*on*, T_8-*on*, *c-m-hrd*, *c-m-res*
$w_1 c_0$	$\bar{c}$-*sa-0*, T_8-*open*,
$w_1 c_1$	$\bar{c}$-*sa-1*, *s-sa-0*, $\bar{s}$-*sa-1*, T_6-*on*, T_7-*on*, T_9-*on*, $\bar{c}$-*m-hrd*, *m-w-hrd*, *m-w-res*
$c_1 c_0$	*m-sa-0*, *m-sa-1*, T_9-*open*, *s-*$\bar{s}$*-hrd*
$w_{00} w_{\times 0} c_{00}$	$\bar{b}'$-*b-hrd*
$w_{11} c_{0\times}$	$\bar{c}'$-*c-hrd*

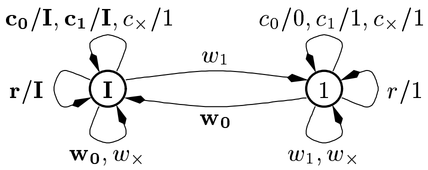


Figure 7: T_3-*on* – an example of a fault that is not reliably testable.

Table 3: Summary of faults that require parametric tests.

Fault Category	Faults
Transistor Faults	T_3-*on*, T_4-*on*
Intra-cell Bridging Faults	b-$\bar{b}$-*hrd*, b-$\bar{b}$-*hrd*, b-$\bar{b}$-*res*, *b*-*w*-*res*, $\bar{b}$-*w*-*res*, *b*-*m*-*res*, $\bar{b}$-*m*-*res*, c-$\bar{c}$-*res*, *c*-*w*-*res*, $\bar{c}$-*w*-*res*, s-$\bar{s}$-*res*
Inter-cell Bridging Faults	$\bar{b}'$-*b*-*res*, $\bar{c}'$-*c*-*res*

Details on the faulty behaviors of this cell can be found in [26]. A summary of the results is given below:

- All input and state stuck-at faults are reliably detectable by functional tests.
- $\frac{2nl}{18nl} \times 100\%$ (approx. 11%) of transistor faults are not detectable by functional CAM tests; however all transistor-open faults that compromise data retention in static memory cells can be reliably detected, due to the enhanced functionality of this type of memory.
- $\frac{9nl+7l-2}{18nl+8l+2n-4} \times 100\%$ (approx. 50%) of faults in the bridging model are not reliably detectable by functional tests.
- Faults that are not detectable by functional tests cause an increase of I_{DD} or I_{DDQ}; hence they are detectable through parametric testing.
- 'Read' operations are inadequate sources of output for testing static CMOS CAMs; 'compare' operations are a more reliable choice.

8 Test Generation

Partly with the aid of the OBSERVER[4] program, we have found a test

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

of length 9 that detects all independently testable faults in a single cell.

This test is then modified into a march test for the n-word by 1-bit CAM:

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

The notation used here is derived from that of A. J. van de Goor [8]; however we have modified it to account for distinct row-wise and column-wise operations, to provide the ability to identify specific rows, columns and cells and to make it more compact in order to extend it to word-oriented memories. The symbol $(\ )^\updownarrow$ denotes n operations. These can be done in order either from n to 1 or from 1 to n; hence the $\updownarrow$. The symbol $(\ )^{n\downarrow 1}$ denotes the direction of the march elements: from row n to row 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.

The reader should note that the $T_{n\text{-}bit}$ test only detects independently testable faults.

8.1 Testing dependently testable faults

As indicated in the previous section, *w-sa-1* is a dependently testable fault. Since the FSMs of the faulty and fault-free cell are identical, no elementary test exists. However, this cell is always accessible due to the faulty w; hence operations on some other fault-free cell will also affect the faulty cell. Since this fault affects all the cells along the faulty w, the second cell must be connected to a different w. The fault can be detected in the following manner:

[4]OBSERVER is a program for diagnosing and testing sequential machines. It is based on the theory developed in [5] and was written at the University of Waterloo by C.-J. Shi; additional features were later added by P. Kwiatkowski and P. R. Sidorowicz.

Assume w^i-*sa-1*. Since word i is always accessed, a 'read' or 'write' operation 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 $1 \leq u < n$. Initially, 0s are written into every word. If word u is the faulty word, $u \neq n$, then w_1^n will write 1s 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, as the location of the faulty word is known to be n. Thus, the occurrence of a multiple hit indicates the *w-sa-1* fault.

8.2 Complete Test for n-word by 1-bit CAM

To detect both the independently and dependently testable faults in our fault model, we combine $T_{n\text{-}bit}$ with $T_{w\text{-}sa\text{-}1}$:

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

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

8.3 Test for 1-word by l-bit CAM

Suppose j is some cell in the l-bit CAM word. During a 'compare' operation a mismatch in a single cell j suffices to mismatch the entire word. Most of the identified faults manifest themselves with mismatches, and hence can be detected by performing the respective tests on all cells in parallel. The exceptions are: c^j-*sa-0*, $\bar{c}^j$-*sa-0*, T_7^j-*open*, T_8^j-*open*, b^j-*m-hrd*, $\bar{b}^j$-*m-hrd*, c^j-$\bar{c}^j$-*hrd* and $\bar{c}^{(j+1)}$-c^j-*hrd*.

Detection of each of these faults, where $1 \leq j \leq l$, 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 m. To propagate a faulty response along m, 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. In this manner one c_0 and $(l-1)$ operations $c_\times$ are performed on each cell. Also, each pair of cells undergoes a $c_{0\times}$, necessary for the detection of the $\bar{c}^{(j+1)}$-c^j-*hrd* faults.

The inter-cell fault $\bar{c}^{(j+1)}$-c^j-*hrd* requires a special pattern to be used during a 'write' operation. We address this issue by inserting a $w_{\times 0\ldots\times 0}c_{0\ldots 0}$ followed by a $w_{0\times\ldots 0\times}c_{0\ldots 0}$ to detect all possible $\bar{c}^{(j+1)}$-c^j-*hrd* faults.

The extension of the T_{cell} test to the 1-word by l-bit CAM takes the form

$$\begin{aligned} T_{word} &= w_{0\ldots 0}w_{1\ldots 1}w_{\times\ldots\times}c_{1\ldots 1}[c_{\times^{j-1}0\times^{l-j}}]^{\leftrightarrow} \\ &\quad w_{0\ldots 0}w_{\times\ldots\times}c_{0\ldots 0}[c_{\times^{j-1}1\times^{l-j}}]^{\leftrightarrow} \\ &\quad (w_{\times 0\ldots\times 0}c_{0\ldots 0}w_{0\times\ldots 0\times}c_{0\ldots 0}), \end{aligned}$$

where $w_{0\ldots 0}$ denotes the writing of an all-0 word, $w_{\times 0\ldots\times 0}$ denotes writing 0s to odd bits within a word and masking the even bits, etc.

8.4 Test for n-word by l-bit CAM

The combination of both extensions described above yields the specification for the behavior of a n-word by l-bit CAM.

The following test detects all reliably detectable faults under the input stuck-at, state stuck-at, transistor stuck-(on/open) and bridging fault models:

$$\begin{aligned} T_{S\&B} &= (w^i_{0\ldots 0})^{\updownarrow} \\ &\quad (w^i_{1\ldots 1}w^i_{\times\ldots\times}c_{1\ldots 1})^{n\downarrow 1}[c_{\times^{j-1}0\times^{l-j}}]^{\leftrightarrow} \\ &\quad (w^i_{0\ldots 0}w^i_{\times\ldots\times}c_{0\ldots 0})^{n\downarrow 1}[c_{\times^{j-1}1\times^{l-j}}]^{\leftrightarrow} \\ &\quad (w^n_{\times 0\ldots\times 0}c_{0\ldots 0}w^n_{0\times\ldots 0\times}c_{0\ldots 0}) \\ &\quad (w^n_{1\ldots 1}c_{1\ldots 1}w^n_{0\ldots 0}w^u_{1\ldots 1}c_{1\ldots 1}). \end{aligned}$$

This test consists of two parts, analogous to those of the $T_{n\text{-}bit\text{-}compl}$ test. The first part consists of an initialization which sets all words to $0\ldots 0$, and two march elements. For a fault-free CAM, each of the march elements should result in n hits with $k = i$, followed by l mismatches at $k = 0$. Any other response indicates a fault. In the second part (last two lines), two hits are expected with $k = 1$, after which multiple hits are monitored, since they indicate faults.

The $T_{S\&B}$ test consists of $5n + 5$ 'write' operations and $2n + 2l + 4$ 'compare' operations. Therefore, the length of our test for all the reliably detectable faults in an n-word by l-bit CAM is $7n + 2l + 9$.

9 Test Evaluation

Several algorithms for testing CAMs have been reported [7, 13, 15, 16, 27]. The tests in [7, 13, 15, 27] are applicable to CAM designs which incorporate an explicit word addressing scheme of the type found in conventional RAMs. The test of [16] imposes extra requirements on the addressing scheme which are not available in most CAMs. It also assumes that 'read' operations do not affect the state of a cell; this assumption is violated by some faults in our fault model. For these reasons we exclude this test from further consideration.

We show how our framework can be used to determine fault coverage with respect to a particular fault model. We illustrate this by establishing the fault coverage of the test reported by Giles and Hunter [7] with respect to the *input stuck-at* fault model of the CAM circuit of Figure 2, under the single-fault assumption. Details can be found in [29].

We represent this test using our notation. Since we index the words in our CAM from 1 to n, (the former being the top, and the latter the bottom of the CAM array) the value of k written into each word i is equal to $i - 1$; $\bar{k}$ denotes the 1's complement of k.

$$
\begin{aligned}
T_{G\&H} \quad = \quad & (w_k^i)^{1\downarrow n}(c_k)^{n\uparrow 1} \\
& (w_{\bar{k}}^i)^{1\downarrow n}(c_{\bar{k}})^{n\uparrow 1} \\
& (w_k^i)^{n\uparrow 1}(c_k)^{1\downarrow n} \\
& (w_{0\ldots 0}^i)^{\updownarrow}[c_{0^{j-1}10^{l-j}}]^{\leftrightarrow} \\
& (w_{1\ldots 1}^i)^{\updownarrow}[c_{1^{j-1}01^{l-j}}]^{\leftrightarrow}.
\end{aligned}
$$

The $T_{G\&H}$ test is limited to CAM arrays where $n \leq 2^l$ in order to ensure that a distinct bit pattern is written into every word of the array. In arrays where $n > 2^l$, duplicate bit patterns would be unavoidable. These duplicates would cause multiple hits, thus precluding the verification of individual match lines, and also precluding the detection of the *w-sa-1* faults.

For the sake of simplicity we assume an n-word by l-bit CAM where $n = 2^l$. In this manner, the address of word (row index) is used as the unique bit-pattern. Whenever necessary, we comment on CAMs where $n < 2^l$, as the unique bit-pattern constraint is also satisfied in this case.

We represent $T_{G\&H}$ from the perspective of an arbitrary cell located at coordinates (i, j) in the array where $1 \leq i \leq n$ and $0 \leq j \leq l-1$. Variable p is a row index (and, coincidentally, the decimal representation of the word content).

$$\begin{aligned} T^{i,j}_{G\&H} \;=\; & w^{i,j}_{((i-1)\,\mathrm{div}\,2^j)\bmod 2}\ [c^j_{(p\,\mathrm{div}\,2^j)\bmod 2}]^{0}_{p=n-1} && (1)\\ & w^{i,j}_{1-(((i-1)\,\mathrm{div}\,2^j)\bmod 2)}\ [c^j_{1-((p\,\mathrm{div}\,2^j)\bmod 2)}]^{0}_{p=n-1} && (2)\\ & w^{i,j}_{((i-1)\,\mathrm{div}\,2^j)\bmod 2}\ [c^j_{(p\,\mathrm{div}\,2^j)\bmod 2}]^{n-1}_{p=0} && (3)\\ & w^{i,j}_0\ [c^j_{I[p,j]}]^{l-1}_{p=0} && (4)\\ & w^{i,j}_1\ [c^j_{1-I[p,j]}]^{l-1}_{p=0} && (5) \end{aligned}$$

where $I[\,]$ is a $n \times l$ identity matrix.

We have shown in [27] that 'compare' operations, even when faulty, do not affect the state of the cell, so interleaving 'write' operations with an arbitrary number of 'compare' operations will not influence any state transitions resulting from 'write' operations. Each cell (i, j), therefore, is subject to one of the following two sequences of write operations: $w_0w_1w_0w_0w_1$ or $w_1w_0w_1w_0w_1$. In the case of $n = 2^l$, each of the initial three 'write' operations is followed by $n/2$ 'compare 0' operations and $n/2$ 'compare 1' operations. The order of 'compare' operations depends on the column j in which the given cell is located, but can be treated as arbitrary. If $n < 2^l$, in the worst case, each of the initial three 'write' operations will only be followed by a sequence of matching 'compare' operations, i.e., a $w^{i,j}_0$ would precede a sequence of c^j_0's or a $w^{i,j}_1$ would precede a sequence of c^j_1's. The last two 'write' operations are always followed by $l-1$ mismatching 'compare' operations, where a w_0 precedes a c_1 and vice-versa, and a single matching 'compare' operation, for all possible values of n and l. The order of occurrence of the single matching 'compare' operation depends on the column j in which the given cell is located.

A sample verification of the presence of an elementary test is given below:

b-sa-0 According to Table 2, a test that detects this fault is $w_0w_\times c_0$. The sequence $w_0^{i,j} w_\times^{i,j}$ does not occur in $T^{i,j}_{G\&H}$. Since after the initial $w_0^{i,j}$ the cell finds itself in an indeterminate state, none of the subsequent 'compare' operations can result in a dependable output.

The above analysis of the $T_{G\&H}$ test shows that, in the cell of Figure 2, $T_{G\&H}$ does not reliably detect the *b-sa-0* and $\bar{b}$*-sa-0* faults. There are $2l$ *b-sa-0* and $\bar{b}$*-sa-0* faults in an n-word by l-bit CAM. Since there are $4n + 8l$ possible single faults in our fault model, $T_{G\&H}$ detects only $(1 - \frac{l}{2n+4l}) \cdot 100\%$ of faults in the input stuck-at fault model. For example [13], for $n = 32$, $l = 29$ only 83.89% of faults are detected.

We can modify the $T_{G\&H}$ test to achieve 100% fault coverage, under the assumption that $n \leq 2^l$. Since in the presence of the *b-sa-0* and $\bar{b}$*-sa-0* faults, a $w_\times$ forces the cell to a determinate but erroneous state, judicious insertion of $w_\times$ into $T_{G\&H}$ will assure detection of these faults. This augmented test, presented below, has length $11n + 2l$.

$$\begin{aligned}
T'_{G\&H} \quad = \quad & (w_k^i w_\times^i)^{1\downarrow n} (c_k)^{n\uparrow 1} \\
& (w_{\bar{k}}^i w_\times^i)^{1\downarrow n} (c_{\bar{k}})^{n\uparrow 1} \\
& (w_k^i w_\times^i)^{1\uparrow n} (c_k)^{n\downarrow 1} \\
& (w_{0\ldots 0}^i)^{\updownarrow} [c_{0^{j-1}10^{l-j}}]^{\leftrightarrow} \\
& (w_{1\ldots 1}^i)^{\updownarrow} [c_{1^{j-1}01^{l-j}}]^{\leftrightarrow}.
\end{aligned}$$

10 Summary

We have presented a formal framework for modeling and testing special-purpose memories. Our goal was to standardize fault models and to associate them with the specific design, functionality and the underlying technology of memories. We strove to systematize the process of fault modeling, test development, evaluation and verification.

We presented our framework using a special-purpose CMOS CAM cell as an example. For this CAM, we have considered fault types obtained from a circuit-level analysis: input stuck-at, state stuck-at, transistor stuck-(on/open), and bridging; for details see [25, 28]. Although some of the considered faults resemble well known functional faults (i.e., cell stuck-at, transition and coupling), many have a distinct behavior. We have shown that for CMOS memories functional tests may not reliably detect up to 50% of faults, and that parametric tests are necessary to obtain full coverage.

We have presented the construction of the $T_{S\&B}$ test, of length $7n+2l+9$, which detects all reliably testable faults in the considered CAM cell, under our fault models. The length of this test can be further reduced by means of design-for-testability (DFT) enhancements [26].

We have also shown how evaluation or verification of tests can be conducted within our framework. We have demonstrated that the $T_{G\&H}$ test originally developed for a different CAM cell [7], does not reliably detect the *b-sa-0* and $\bar{b}$*-sa-0* faults in the input stuck-at fault model. This test can be modified to achieve 100% fault coverage at the cost of increased length. Although we have evaluated this test only for input stuck-at faults, the results obtained indicate the necessity of utilizing fault models that reflect idiosyncrasies of cell designs.

This work has also opened several new avenues for future study. Application of this framework to other types of memories and existing tests, extraction of fault models from circuit simulations and the development of more efficient test generation software are some of the possibilities.

Acknowledgments

We thank Ken Schultz formerly of Nortel Corporation's Memory Development Team and Manoj Sachdev of the Department of Electrical and Computer Engineering at University of Waterloo for providing extensive information regarding various defects and their effects on the behavior of a CAM. We also express our gratitude to all the members of the Maveric Group at the University of Waterloo for their insightful comments and suggestions.

References

[1] S. J. Adams, M. J. Irwin, and R. M. Owens. A parallel, general purpose CAM architecture. In *Proc. 4th MIT Conf. Advanced Research in VLSI*, pages 51–71. MIT Press, 1986.

[2] M. Akata, S. Karube, T. Sakamoto, T. Saito, S. Wakasugi, S. Yoshida, H. Matsuno, and H. Shibata. A scheduling content-addressable memory for ATM space-division switch control. In *International Solid State Circuits Conference*, pages 244–245. IEEE Computer Society Press, 1991.

[3] W. K. Al-Assadi, A. P. Jayasumana, and Y. K. Malaiya. On fault modelling and testing of content-addressable memories. In *Records of the IEEE Workshop on Memory Technology, Design and Testing*, pages 78–83. IEEE Computer Society Press, August 1994.

[4] 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.

[5] 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.

[6] J. D. Garside, S. Temple, and R. Mehra. The AMULET2e cache system. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 208–217. IEEE Computer Society Press, March 1996.

[7] G. Giles and C. Hunter. A methodology for testing content-addressable memories. In *Proc. International Test Conference*, pages 471–474. IEEE Computer Society Press, 1985.

[8] A. J. van de Goor. *Testing Semiconductor Memories.* John Wiley & Sons, 1991.

[9] A. J. van de Goor and S. Hamdioui. Fault models and tests for two-port memories. In *IEEE VLSI Test Symposium*, pages 401–410, Monterrey, CA, April 1998. IEEE Computer Society Press.

[10] T. Hanyu, S. Aragaki, and T. Higuchi. Functionally separated, multiple-valued content-addressable memory and its applications. *IEE Proceedings, Part G, Circuits, Devices and Systems*, 142(3):165–172, June 1995.

[11] S. M. S. Jalaleddine and L. G. Johnson. Associative memory integrated circuit based on neural mutual inhibition. *IEE Proceedings, Part G, Circuits, Devices and Systems*, 139(4):445–449, August 1992.

[12] H. Kadota, J. Miyake, Y. Nishimichi, H. Kudoh, and K. Kagawa. An 8-kbit content-addressable and reentrant memory. *IEEE Journal of Solid-State Circuits*, sc-20(5):951–957, October 1985.

[13] 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.

[14] D. Lamet and J. F. Frenzel. Defect-tolerant cache memory design. In *IEEE VLSI Test Symposium*, pages 159–163. IEEE Computer Society Press, 1994.

[15] K.-J Lin and C.-W. Wu. Functional testing of content-addressable memories. In *Records of the IEEE Workshop on Memory Technology, Design and Testing*, pages 70–75, San Jose, CA, August 1998. IEEE Computer Society Press.

[16] P. Mazumder, J. H. Patel, and W. K. Fucks. Design and algorithms for parallel testing of random access and content addressable memories. In *Proc. ACM/IEEE Design Automation Conference*, pages 688–694. IEEE Computer Society Press, 1987.

[17] P. Mazumder, J. H. Patel, and W. K. Fucks. Methodologies for testing embedded content addressable memories. *IEEE Transactions on Computer-Aided Design*, 7(1):11–20, January 1988.

[18] A. J. McAuley and C. J. Cotton. A self-testing reconfigurable CAM. *IEEE Journal of Solid-State Circuits*, 26(3):257–261, March 1991.

[19] J. L. Mundy, J. F. Burgess, R. E. Joynson, and C. Neugebauer. Low-cost associative memory. *IEEE Journal of Solid-State Circuits*, SC-7(5):364–369, October 1972.

[20] T. Ogura, S. Yamada, and T. Nikaido. A 4-kbit associative memory LSI. *IEEE Journal of Solid-State Circuits*, SC-20(6):1277–1282, December 1985.

[21] T. Ogura, S. Yamada, and M. Tan-no. A 20-kbit associative memory LSI for artificial intelligence machines. *IEEE Journal of Solid-State Circuits*, 24(4):1014–1020, August 1989.

[22] M. Sachdev. *Defect Oriented Testing for CMOS Analog and Digital Circuits*. Kluwer Academic Publishers, 1998.

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

[24] Y-C. Shin, R. Sridhar, V. Demjanenko, P. W. Palumbo, and S. N. Srihari. A special-purpose content-addressable memory chip for real-time image processing. *IEEE Journal of Solid-State Circuits*, 27(5):737–744, May 1992.

[25] P. R. Sidorowicz. Modeling and testing transistor faults in content-addressable memories. In *Records of the IEEE Workshop on Memory Technology, Design and Testing*, pages 83–90, San Jose, CA, August 1999. IEEE Computer Society Press.

[26] P. R. Sidorowicz. *A Formal Framework for Modeling and Testing Memories.* PhD thesis, University of Waterloo, May 2000.

[27] P. R. Sidorowicz and J. A. Brzozowski. An approach to modeling and testing memories and its application to CAMs. In *IEEE VLSI Test Symposium*, pages 411–416. IEEE Computer Society Press, April 1998.

[28] 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.

[29] P. R. Sidorowicz and J. A. Brzozowski. Verification of CAM tests for input stuck-at faults. In *Records of the IEEE Workshop on Memory Technology, Design and Testing*, pages 76–82. IEEE Computer Society Press, August 1998.

[30] L. R. Tamura, T-S. Yang, D. E. Wingard, M. A. Horowitz, and B. A. Wooley. A 4-ns BiCMOS translation-lookaside buffer. *IEEE Journal of Solid-State Circuits*, 25(5):1093–1101, October 1990.

[31] J. C. Wade and C. G. Sodini. Dynamic cross-coupled bit-line content-addressable memory cell for high-density arrays. *IEEE Journal of Solid-State Circuits*, SC-22(1):119–121, February 1987.

[32] T. Yamagata, M. Mihara, T. Hamamoto, Y. Murai, T. Kobayashi, and M. Yamada. A 288-kb fully parallel content addressable memory using a stacked-capacitor cell structure. *IEEE Journal of Solid-State Circuits*, 27(12):1927–1933, December 1992.