Evaluating Tests for Input Stuck-at Faults in Word-Oriented Static Random-Access Memories*

Piotr R. Sidorowicz

Department of Computer Science
University of Waterloo
Waterloo, Ontario, N2L 3G1
Canada

Abstract

An evaluation of well-known tests: MATS+, MATS++, MARCH Y and MARCH C- with respect to input stuck-at faults in a n-word by l-bit static CMOS random-access memory (SRAM) array is presented. First, an SRAM cell's behavior is analysed at the transistor-network, event-sequence, and finite-state machine (FSM) level. Then, an input stuck-at fault model for an SRAM is defined. We show that the word oriented extensions of the above-mentioned tests detect reliably at most $\frac{n+4l}{2n+4l} \cdot 100\%$ of faults in our fault model, which for large n constitutes roughly 50% of faults. We propose a DFT enhancement that would increase this coverage to 100%.

Keywords. CMOS, design for testability, fault modeling, MARCH C-, MARCH Y, MATS+, MATS++, SRAM, stuck-at faults, testing.

1 Introduction

Word-oriented static random-access memories are well-known storage devices. Though their bit densities are not nearly as great as those of DRAMs, their speed and reliability makes them currently the proper choice for embedding in larger ASICs. One application of this type of memory is the implementation of the data field in caches.

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

In this report we investigate the testability properties of a word-oriented SRAMs based on the cell shown in Figure 1(a).

The formal approach used in this report has been originally developed for modeling faults in content-addressable memories (CAMs) [5], and it has been successfuly utilized to evaluate exsisting CAM tests [6]. The *input stuck-at fault model*, also introduced in [5], consists of any stuck-at fault which affects the input lines of a memory cell.

The focus of this report is the evaluation of well-known tests, MATS+, MATS++, MARCH Y and MARCH C-, with respect to the input stuck-at fault model of an n-word by l-bit SRAM. Here, we demonstrate that any test that uses 'read' and 'write' operations can reliably detect at most 50% of the faults in our fault model. We also show that MATS+ has even worse fault coverage. Finally, we propose a hardware modification that would allow for 100% fault coverage.

The remainder of this report is organized as follows: An analysis of an SRAM cell's behavior at the transistor-network, event-sequence and FSM represenation levels is presented in Section 2. The fault model is presented in Section 3. In Section 4 the MATS+, MATS++, MARCH Y and MARCH C- tests are evaluated. A design-for-testability modification facilitating an improvement of the fault coverage is proposed in Section 5. Section 6 contains some concluding remarks.

2 Analysis of an SRAM cell

The following behavioral analysis has been previously applied to CAMs. Since an SRAM cell constitutes only the storage section of a CAM cell, a simplified analysis is presented here.

2.1 Transistor-Network Level

The circuit of the cell is shown in Figure 1(a). It consists of 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. Changes of the cell's state, when writing a 1 to a cell containing a 0, for example, are a result of the dominant influence of stronger pull-down transistors. (Weaker pull-up transistors maintain quiescent states

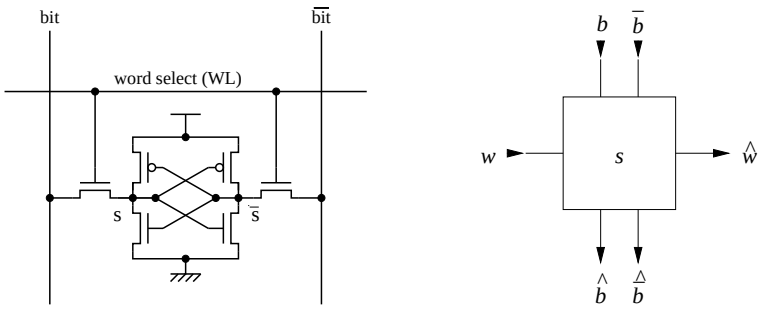

Figure 1: (a) SRAM cell, (b) its model.

of the cell.) 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 resultant voltage differential between the $bit/\overline{bit}$ lines is detected by sense amplifiers that re-create the contents of the accessed word.

Altogether, three operations can be performed on a SRAM cell: 'write 0', 'write 1' and 'read'.

2.2 Event-Sequence Level

To model input stuck-at faults, we have to understand how they affect the internal operation of the cell. Since no clock signal is supplied to individual cells, a single SRAM 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 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 for $bit/\overline{bit}$, WL. Now, b and $\bar{b}$ are inputs and $\hat{b}$ and $\hat{\bar{b}}$ are outputs. Although WL is not usually meant to provide any output, monitoring the state of this line, if possible, might improve the SRAM's testability. For this reason, we generalize our model to include this line; w for input, and $\hat{w}$ 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 7-tuple:

$$C_T = (w, b, \bar{b}, s, \hat{w}, \hat{b}, \hat{\bar{b}}).$$

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 for simplicity. The simplified total state is symbolically

represented by $C = w\ b\bar{b}\ \cdot s$, where the input variables have been separated by a space for readability 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 $Y = \{0, 1, \tilde{0}, \tilde{1}, I\}$. 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 simultaneously conduct. The SRAM cell has two possible initial states: $C = 0\ 11 \cdot 0$ and $C = 0\ 11 \cdot 1$.

Table 1: Read and write operations in a fault-free SRAM cell.

Read operations		
Description	$(s = 0)$ $w\ b\bar{b}\cdot s$	$(s = 1)$ $w\ b\bar{b}\cdot s$
initial state	$0\ 11\cdot 0$	$0\ 11\cdot 1$
float $bit/\overline{bit}$	$0\ \tilde{1}\tilde{1}\cdot 0$	$0\ \tilde{1}\tilde{1}\cdot 1$
raise WL	$1\ \tilde{1}\tilde{1}\cdot 0$	$1\ \tilde{1}\tilde{1}\cdot 1$
bit or $\overline{bit}$ discharges[1]	$1\ 0\tilde{1}\cdot 0$	$1\ \tilde{1}0\cdot 1$
read $bit/\overline{bit}$ & lower WL	$0\ \tilde{0}\tilde{1}\cdot 0$	$0\ \tilde{1}\tilde{0}\cdot 1$
raise $bit/\overline{bit}$	$0\ 11\cdot 0$	$0\ 11\cdot 1$
Write operations $(s = 0)$		
Description	w_0 $w\ b\bar{b}\cdot s$	w_1 $w\ b\bar{b}\cdot s$
initial state	$0\ 11\cdot 0$	$0\ 11\cdot 0$
set $bit/\overline{bit}$	$0\ 01\cdot 0$	$0\ 10\cdot 0$
raise WL	$1\ 01\cdot 0$	$1\ 10\cdot 0$
new state	$1\ 01\cdot 0$	$1\ 10\cdot 1$
lower WL	$0\ 01\cdot 0$	$0\ 10\cdot 1$
raise $bit/\overline{bit}$	$0\ 11\cdot 0$	$0\ 11\cdot 1$
Write operations $(s = 1)$		
Description	w_0 $w\ b\bar{b}\cdot s$	w_1 $w\ b\bar{b}\cdot s$
initial state	$0\ 11\cdot 1$	$0\ 11\cdot 1$
set $bit/\overline{bit}$	$0\ 01\cdot 1$	$0\ 10\cdot 1$
raise WL	$1\ 01\cdot 1$	$1\ 10\cdot 1$
new state	$1\ 01\cdot 0$	$1\ 10\cdot 1$
lower WL	$0\ 01\cdot 0$	$0\ 10\cdot 1$
raise $bit/\overline{bit}$	$0\ 11\cdot 0$	$0\ 11\cdot 1$

The behavior of a cell is represented by sequences of events that take place during the three operations. Table 1 lists events that occur during

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

these operations. 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 each operation starts in (top entry) and returns to (bottom entry) an initial state.

Example: 'Write 0' operation.

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

If it were possible to control any input and observe any output, testing would be a trivial process: if a simple comparison of the state of each line to its expected value were made, detection of any disagreement would indicate a fault. Unfortunately, it is not possible to monitor any real circuit at this level of detail and, thus, a more abstract model of an SRAM cell is needed.

2.3 FSM Representation Level

Most testing algorithms utilize sequences of 'read' and 'write' operations as input, and observe the resulting output. Accordingly, we introduce a sequential machine model of an SRAM cell. This model is derived from the event-sequence model of Section 2.2.

We represent the behavior of a fault-free SRAM cell as an FSM, which is shown in Figure 2(a). The input x of this FSM comprises all three oper-

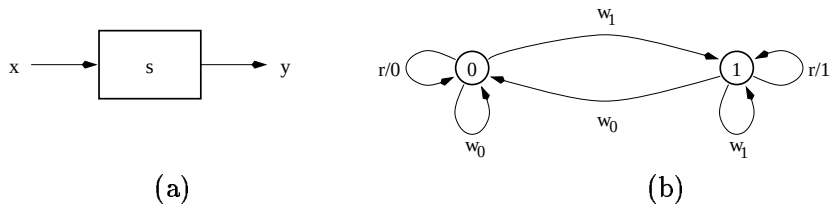

Figure 2: (a) Simplified behavioral model (b) Behavior of a fault-free cell.

ations of a SRAM cell, and the output y comprises the responses to these operations. State s represents the value stored by the cell. Formally, a *fault-free SRAM cell* is a Mealy sequential machine $M = (Q, X, Y, \delta, \lambda)$, where $Q = \{0, 1\}$ is the set of states, $X = \{r, w_0, w_1\}$ 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

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

This FSM 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 $\lambda(0, w_1) = \$$.

In an *faulty SRAM cell* FSM, the faulty set of states $Q' = Q \cup \{I\}$ and the faulty set of output symbols $Y' = Y \cup \{I\}$, where I represents an "indeterminate" logic value.

3 Input Stuck-at Faults in an SRAM cell

A event-sequence analysis has been performed for all six input stuck-at faults under a single-fault assumption; these analyses are presented in Appendix A. Subsequently, FSM models for each of these faults have been developed. The resulting *faulty SRAM cells* are presented in Figure 3, where incorrect operations and outputs are in bold type.

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

From the event-sequence model we construct an FSM. In this example, operations w_0 and r are incorrect, but w_1 is correct; hence we get Figure 3(c).

The reader should note that the *b-sa-0* and $\bar{b}$*-sa-0* faults increase the state set Q by the "indeterminate" state I. This state can be interpreted in two ways. From the "asynchronous" perspective it represents a temporary, metastable state, where the cell holds some indeterminate logic value. This interpretation is important when dealing with multi-port SRAMs, where simultaneous operations are possible. From the "synchronous" standpoint it represents the loss of information about the current state of the cell, since the outcome of metastable states is non-deterministic. In either case, state I is not considered as an initial state.

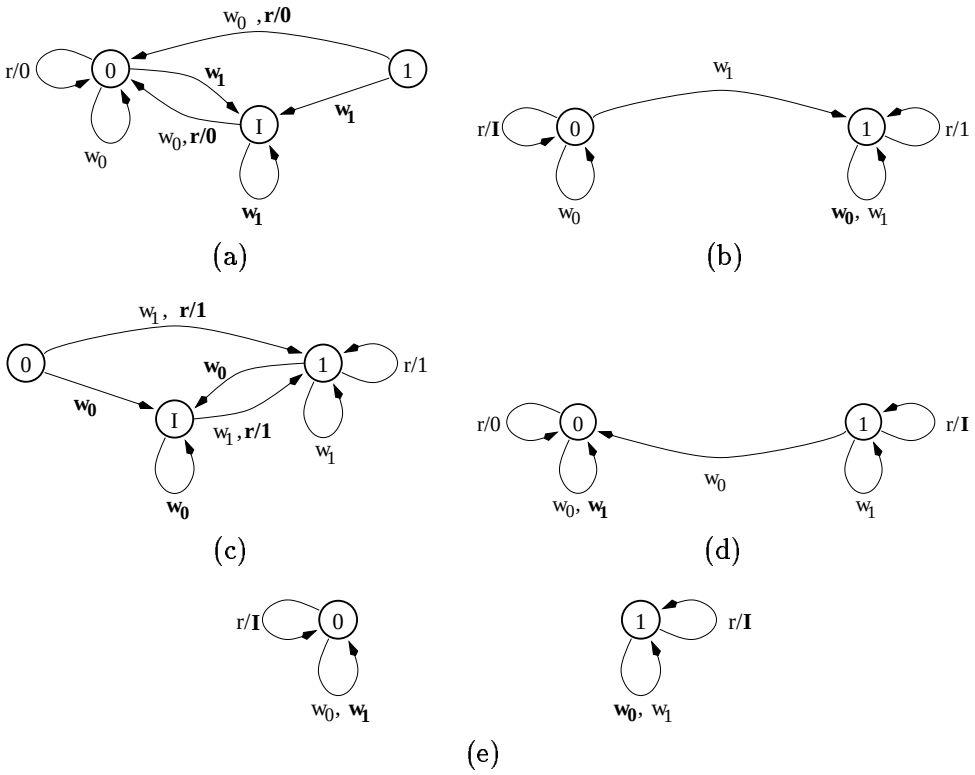

Figure 3: Faulty behaviors of a SRAM cell: (a) *b-sa-0*, (b) *b-sa-1*, (c) $\bar{b}$*-sa-0*, (d) $\bar{b}$*-sa-1*, (e) *w-sa-0*.

The comparison of each of the faulty machines to the fault-free SRAM cell has led to the derivation of simple tests for four out of six possible faults. We refer to the shortest test that detects a particular fault in a single cell as an *elementary test*; these tests are essential to the detection of the associated faults. Table 2 lists all the elementary tests for an SRAM cell.

These tests have been generated by the OBSERVER[2] program for all input stuck-at faults except the *w-sa-1* and the *w-sa-0* faults. We refer to the faults that are detectable in a single cell as *independently testable*.

The reader should note that in the presence of *b-sa-1*, $\bar{b}$*-sa-1* and *w-sa-0* faults the 'read' operation produces an unreliable output, and also that in

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

Table 2: Summary of input stuck-at faults and elementary tests.

Fault	Elementary Test	Faulty Response	Fault-Free Response
b-sa-0	$w_1 r$	0	1
b-sa-1	$w_1 w_0 r$	1	0
$\bar{b}$*-sa-0*	$w_0 r$	1	0
$\bar{b}$*-sa-1*	$w_0 w_1 r$	0	1
w-sa-0	not testable reliably		
w-sa-1	see Section 3.1.2		

the presence of the *b-sa-0* and $\bar{b}$*-sa-0* faults this operation changes the state of the cell. For instance, in the last example a faulty w_0 forces the cell to an indeterminate state I, whereas a good cell would remain in state 0. This means that a faulty w_0 does not necessarily sensitize this fault. Fortunately, the faulty r forces the cell to state 1 and provides a reliable faulty output 1.

The *w-sa-0* fault is an instance of an *address decoder fault A* [3] or a *stuck-open fault* [2]. Also, when started in state 0 (state 1) the *w-sa-0* fault is superficially similar to a cell stuck-at-0 (stuck-at-1) fault, however, the key difference between these faults is that in the case of the *w-sa-0* fault the 'read' operations produce an indeterminate faulty output I. Memory designers often claim that this indeterminate output is resolved preferentially due to inherently unequal bias of sense amplifiers, and that this fault should manifest itself as one of the cell stuck-at faults; we, nevertheless, regard this input stuck-at fault as generally not testable.

The *w-sa-1* fault FSM, is identical to that of the fault-free SRAM cell; therefore, no test exists for a single cell. This fault, however, is a generalization of an *address decoder fault D* [3] (a multiple coupling fault), where operations on any other word in the memory will affect the faulty word. Therefore, this fault is detectable in conjunction with other words [3]; this will be considered later. Consequently, the *w-sa-1* fault is the only input stuck-at fault that is not independently testable.

The *b-sa-1* ($\bar{b}$*-sa-1*) fault is similar to a cell unable to undergo the $1 \rightarrow 0$ ($0 \rightarrow 1$) transition. This fault is detected by $w_1 w_0 r$ ($w_0 w_1 r$).

3.1 Extension to n-word by 1-bit SRAM

We now consider an n-bit SRAM where each word consists of a single cell (a bit-oriented memory). At this point, an assumption must be made about the functionality of the peripheral circuitry in a fault-free SRAM.

1. The address decoder can raise at most a single word line. Consequently, only one word can be written to or read from at any time.

We model the correct behavior of an n-word by 1-bit SRAM as a Mealy sequential machine $M = (Q, X, Y, \delta, \lambda)$, where $Q = \{0,1\}^n$, $X = (\bigcup_{i=1}^{n} A_i)$ with $A_i = \{r^i, w_0^i, w_1^i\}$, $Y = \{0, 1, \$\}$ and the transition function δ and the output function λ are defined by

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

and

$$\lambda((q^1, \ldots, q^i, \ldots, q^n), x) = \begin{cases} q^i, & \text{if } x = r^i, \\ \$, & \text{otherwise.} \end{cases}$$

The inputs r^i, w_0^i, w_1^i denote the 'read', 'write 0' and 'write 1' operations on word i, respectively. As before, the output \$ is merely a formal symbol denoting lack of output.

3.1.1 The w^i-*sa-0* fault

As stated before, in the presence of a w^i-*sa-0* fault none of the cells along the faulty word line can be written to, or reliably read. Moreover, no operation on any of the remaining $n-1$ fault-free words can sensitize the cells along the faulty word line such that this fault could be detected. Therefore, in an n-word by 1-bit SRAM, there are n possible w^i-*sa-0* faults that cannot be reliably detected by any combination of 'read' and 'write' operations.

3.1.2 Testing the w^i-*sa-1* fault

As indicated earlier, w^i-*sa-1* is not independently testable and affects all the cells along the faulty *WL*. However, this fault is detectable in conjunction with another memory word in the following manner:

Assume w^i-*sa-1*. Since word i is always accessed, a 'read' or 'write' operation may be applied to two words simultaneously. In this case, a 'write'

operation is always successful for both words. On the other hand, the result of a 'read' operation will be unreliable if the two accessed words contain opposing values; this will not, however, disrupt the contents of either word.

A possible test is:

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

where $1 \leq i \leq n$. 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 $(\ )^{(n-1)\downarrow 1}$ denotes the direction of the march element: from $n-1$ to 1. Initially, 0s are written into every word. If word u, where $1 \leq u < n$ is the faulty word, then w_1^n will write 1s to both u and n. Thus r^i, when $i = u$, will generate a 1. Now, let $u = 1$. If word n is the faulty word, then word 1 is not, so w_0^1 will write 0 to words 1 and n, and the subsequent r^n will generate a 0. Thus, the occurrence of a 1 during any of the first $n-1$ 'read' operations and an occurrence of a 0 due to the last 'read', indicates the *w-sa-1* fault.

3.2 Extension to a 1-word by l-bit SRAM

We now consider a single row of cells comprising an SRAM word.

Let $\mathcal{B} = \{0, 1\}$, and to extend this set to represent l-bit words, let $\mathcal{V} = \mathcal{B}^l$. The behavior of a 1-word by l-bit SRAM can be specified by a Mealy sequential machine $M = (Q, X, Y, \delta, \lambda)$ where $Q = \mathcal{V}$, $X = \{r\} \cup (\bigcup_{p \in \mathcal{V}} w_p)$, and $Y = \mathcal{V} \cup \{\$\}$. For $q \in \mathcal{V}$, $x \in X$ the transition function δ and the output function λ are defined by

$$\delta(q, x) = \begin{cases} q, & \text{if } x = r, \\ p, & \text{if } x = w_p,\ p \in \mathcal{V}, \end{cases} \quad \text{and} \quad \lambda(q, x) = \begin{cases} q, & \text{if } x = r, \\ \$, & \text{otherwise.} \end{cases}$$

Faults affecting the $bit/\overline{bit}$ lines of each of the l cells are detectable by performing the respective tests on all cells in parallel since these faults manifest themselves with an erroneous l-bit output of the 'read' operation. Detection of the *w-sa-0* fault is not possible, as this fault affects the entire word the same manner.

3.3 Extension to n-word by l-bit SRAM

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

Let $\mathcal{V}$ be defined as before. The correct behavior of a n-word by l-bit SRAM is denoted by a Mealy sequential machine $M = (Q, X, Y, \delta, \lambda)$, where

$Q = \mathcal{V}^n$, $X = (\bigcup_{i=1}^{n} A_i)$ with $A_i = \{r^i\} \cup (\bigcup_{p \in \mathcal{V}} w_p^i)$, $Y = \mathcal{V} \cup \{\$\}$. For $1 \leq i \leq n$ and for $q^i \in \mathcal{V}$, $x \in X$, $p \in \mathcal{V}$, the transition function δ and the output function λ are defined by

$$\delta((q^1, \ldots, q^i, \ldots, q^n), x) = \begin{cases} (q^1, \ldots, q^i, \ldots, q^n), & \text{if } x = r^i, \\ (q^1, \ldots, p, \ldots, q^n), & \text{if } x = w_p^i, \end{cases}$$

and

$$\lambda((q^1, \ldots, q^i, \ldots, q^n), x) = \begin{cases} q^i, & \text{if } x = r^i,\ 1 \leq i \leq n, \\ \$, & \text{otherwise.} \end{cases}$$

From the combined fault analysis we conclude that, for the input stuck-at fault model, only the *w^i-sa-0* faults cannot be tested reliably. Since in a n-word by l-bit SRAM, n such faults may exist, the best possible fault coverage of any test under our fault model is $\frac{n+4l}{2n+4l} \cdot 100\%$, which for an 8k-word by 8-bit SRAM [2] is 50.1%, but for a 32-word by 73-bit SRAM [4] is 91.01%.

4 Evaluation of Tests

In this section we analyse well-known tests with respect to the input stuck-at fault model. A complete evaluation of the MATS+ test is given here. The same approach has been applied to MATS++, MARCH Y and MARCH C- and the summaries of these evaluations are given. The complete analyses of the latter three tests can be found in Appendix B.

4.1 Evaluation of the MATS+ test

The MATS+ test [3] for a n-word by 1-bit SRAM, is of length $5n$, and is presented below:

$$\text{MATS+} = (w_0^i)^{\updownarrow}(r^i w_1^i)^{n\downarrow 1}(r^i w_0^i)^{1\uparrow n}.$$

First, all the words are initialized to 0. Then, in the fault-free SRAM, each w_1^i in the second march element is preceeded by an r^i which should produce a 0. This march element is performed in the direction from n to 1. The rest of the test sequence is similar, with 0 and 1 interchanged and the march element direction reversed.

4.1.1 Evaluation of the MATS+ test for a single cell

First, we restrict the above test to a single cell. Each cell in this test is subject to the following sequence of operations: $w_0 r w_1 r w_0$. Table 3 provides the comparison of the fault-free response with the faulty responses of all independently testable faults. The deviations from the fault-free response are indicated in boldface. Tests for individual faults that are included in MATS+ are also given. From this table it is clear that the *b-sa-1* fault

Table 3: Evaluation of the MATS+ test for a single cell.

Fault	MATS+ $w_0 r w_1 r w_0$	Elementary tests for input stuck-at faults within MATS+
fault-free cell	$-\ 0\ -\ 1\ -$	
b-sa-0	$-\ 0\ -\ \mathbf{0}\ -$	$w_0 r \mathbf{w_1 r} w_0$
b-sa-1	$-\ \mathbf{I}\ -\ 1\ -$	missing test: $w_1 w_0 r$
$\bar{b}$*-sa-0*	$-\ \mathbf{1}\ -\ 1\ -$	$\mathbf{w_0 r} w_1 r w_0$
$\bar{b}$*-sa-1*	$-\ 0\ -\ \mathbf{0}\ -$	$\mathbf{w_0 w_1 r} w_0$

will not be detected reliably, as the elementary test for this fault is not present in the MATS+ sequence for a single cell. The reader can verify that for an inverted MATS+ test $w_1 r w_0 r w_1$, where each cell undergoes the complementary sequence of operations, it is the $\bar{b}$*-sa-1* fault that will not be reliably detected.

4.1.2 Evaluation of the MATS+ test for n-word by 1-bit SRAM

We verify that MATS+ detects w^i*-sa-1* faults by showing that it contains the $T_{w\text{-}sa\text{-}1}$ test. Given

$$\text{MATS+} = (w_0^i)^{\updownarrow}\ (r^i w_1^i)^{n\downarrow 1}\ (r^i w_0^i)^{1\uparrow n},$$

We expand the latter two march elements and get

$$(w_0^i)^{\updownarrow}\ r^n w_1^n (r^i w_1^i)^{(n-1)\downarrow 1}\ r^1 w_0^1 (r^i w_0^i)^{2\uparrow(n-1)} r^n w_0^n.$$

We can disregard the $(w_1^i)^{(n-1)\downarrow 1}$, $(w_0^i)^{2\uparrow(n-1)}$ and w_0^n operations as, for any given word, they occur after a 'read', and they are not required to sensitize the faulty word. The sensitization is accomplished first by the w_1^n and then by the w_0^1. We can also disregard the r^n, r^1 and $(r^i)^{2\uparrow(n-1)}$ operations, as

they do not corrupt the contents of the cells[3] and thus we get

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

which completes our proof.

We have shown that MATS+ is unable to detect *b-sa-1* and w^i*-sa-0* [4] faults. We thus conclude that in a n-word by 1-bit SRAM, under the single-fault assumption, this test can reliably detect $\frac{n+3}{2n+4} \cdot 100\%$ possible input stuck-at faults, which is roughly 50% of faults (for large n) in the input stuck-at model.

4.1.3 Evaluation of the MATS+ test for n-word by l-bit SRAM

The word-oriented extension of the MATS+ test for a n-word by l-bit SRAM takes the form:

$$\text{MATS+} = (w_{0\ldots0}^i)^{\updownarrow} (r^i w_{1\ldots1}^i)^{n\downarrow 1} (r^i w_{0\ldots0}^i)^{1\uparrow n},$$

where $w_{0\ldots0}$ denotes the writing of an all-0 word, etc. This is often referred to as a *data background* [2]. The reader should note that, for input stuck-at faults, this extended test detects the same faults per cell, as the test for a single cell. Changing the data background does not affect the relative number of faults that may be undetected; it does, however, affect the type of the undetectable faults. There are l possible b^j*-sa-1* faults, where $1 \leq j \leq l$ in a 1-word by l-bit SRAM, and therefore, under the single-fault assumption, this word-oriented extension of the MATS+ test will reliably detect $\frac{n+3l}{2n+4l} \cdot 100\%$ of faults under the input stuck-at fault model. Thus, for the previously mentioned (8k $\times$ 8)-bit SRAM, the coverage is 50.05% - a decrease of 0.05% from the optimal coverage of Section 3.3. However, for memories with long words, such as the (32 $\times$ 73)-bit SRAM the coverage is only 70.5% - a decrease of 20.51%.

4.2 Evaluation of the MATS++ test

The MATS++ test [3] is a well-known, bit-oriented test, of length $6n$. One possible word-oriented extension of this test for an n-word by l-bit SRAM is:

$$\text{MATS++} = (w_{0\ldots0}^i)^{\updownarrow} (r^i w_{1\ldots1}^i)^{n\downarrow 1} (r^i w_{0\ldots0}^i r^i)^{1\uparrow n}.$$

[3] Single fault assumption.

[4] See Section 3.1.1

As shown in Appendix B, this word-oriented extension of the MATS++ test will reliably detect all the detectable input stuck-at faults, which constitute $\frac{n+4l}{2n+4l} \cdot 100\%$ of all the faults in the fault model; hence for large n the fault coverage is roughly 50%.

4.3 Evaluation of the MARCH Y test

Another well-known, bit-oriented test is the MARCH Y test [3], of length $8n$. A word-oriented extension of this test is presented below:

$$\text{MARCH Y} = (w^i_{0\ldots0})^{\updownarrow}(r^i w^i_{1\ldots1} r^i)^{n\downarrow 1}(r^i w^i_{0\ldots0} r^i)^{1\uparrow n}(r^i)^{\updownarrow}.$$

This extension of the MARCH Y test will also reliably detect all the detectable input stuck-at faults, i.e. $\frac{n+4l}{2n+4l} \cdot 100\%$ of all the faults in the fault model. This coverage is identical to that of the MATS++, yet the MARCH Y is longer, as it contains redundant elements with respect to the detection of input stuck-at faults.

4.4 Evaluation of the MARCH C- test

The MARCH C- test [3], of length $10n$, is also a well-known, bit-oriented test. A word-oriented extension of this test is presented below:

$$\text{MARCH C-} = (w^i_{0\ldots0})^{\updownarrow}(r^i w^i_{1\ldots1})^{n\downarrow 1}(r^i w^i_{0\ldots0})^{n\downarrow 1}(r^i w^i_{1\ldots1})^{1\uparrow n}(r^i w^i_{0\ldots0})^{1\uparrow n}(r^i)^{\updownarrow}.$$

This extension of the MARCH C- test will also reliably detect all the detectable input stuck-at faults. This coverage of input stuck-at faults is identical to that of the MATS++ and MARCH Y, yet the MARCH C- is longer than either of these two tests.

5 DFT Suggestions

Earlier we have shown that the *w^i-sa-0* faults cannot be tested reliably by any combination of 'read' and 'write' operations. These faults constitute $\frac{n}{2n+4l} \cdot 100\%$ of faults in our fault model. In Section 2.2 we mentioned that considering word select lines as a source of output $\widehat{w}^i$ could improve the SRAM's testability. By providing an additional output $w\text{-}low = NOR(\widehat{w}^1, \ldots, \widehat{w}^n)$, depicted as a logic gate in Figure 4, the fault coverage can be improved significantly. This simple DFT enhancement, by indicating that none of the word lines is active, can directly detect any *w^i-sa-0* fault, increasing the fault coverage of the MATS++, MARCH Y and MARCH C- tests to 100% with respect to the input stuck-at fault model.

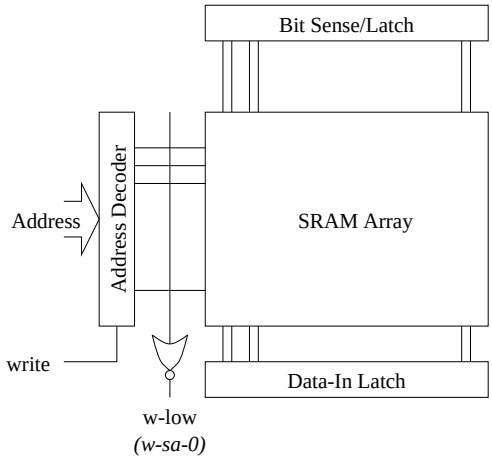

Figure 4: DFT suggestions.

6 Concluding Remarks

Using a n-word by l-bit static CMOS SRAM as a basis, we have evaluated some commonly known tests with respect to a simple class of faults: the input stuck-at faults. We have demonstrated that these tests may fail to detect close to 50% of faults (for large n) in our fault model. This poor performance is attributed to the fact that 'read' operations are insufficient for the detection of all the input stuck-at faults in this type of static CMOS SRAMs. In order to provide 100% fault coverage some DFT enhancement, such as the one that has been suggested, is necessary. We have shown that tests MATS++, MARCH Y and MARCH C- have the same fault coverage despite their different lengths, and that the MATS+ test has an inferior fault coverage in comparison to the previous three. We have also demonstrated that if a column-wise fault (eg. a bit line fault) is not detectable by a particular bit-oriented test, then complementing the data values in the test will result in a complementary fault not being detected. This property is magnified when such a test is modified to word-oriented memories, because different data backgrounds will have a different combination of complementary types of faults being undetected.

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

Acknowledgments

I thank my supervisor Janusz Brzozowski for suggesting the application of our methodology to SRAMs, and for his insightful comments and suggestions regarding this report.

References

[1] 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.

[2] R. Dekker, F. Beenker, and L. Thijssen. A realistic fault model and test algorithms for static random access memories. *IEEE Transactions on Computer-Aided Design*, 9(6):567–572, June 1990.

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

[4] 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.

[5] 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.

[6] 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.

A SRAM Fault Analysis

In this section operations observably affected by input stuck-at faults will be discussed. The behaviors of operations not listed here may differ slightly from their fault-free counterparts in the event-sequence model; however in the FSM model these differences cannot be observed.

***b-sa-0* fault.** This fault does not affect operations where the b line is normally driven to 0. This includes w_0 and r in state 0. Table 4 describes the faulty behaviors in the presence of this fault. Deviations from the fault-free behavior are indicated by bold type.

Table 4: SRAM cell operation with *b-sa-0* fault.

Read operation	
	$(s = 1)$
Description	$w\ b\bar{b}{\cdot}s$
initial state	0 **0**1·1
float $b/\overline{b}$	0 **0**$\tilde{1}$·1
raise w	1 **0**$\tilde{1}$·1
b or $\overline{b}$ discharges	1 **0**$\tilde{1}$·**0**
read $b/\overline{b}$ & lower w	0 **0**$\tilde{1}$·**0**
raise $b/\overline{b}$	0 **01·0**

Write 1 operation	
	$(s = 0)$
Description	$w\ b\bar{b}{\cdot}s$
initial state	0 **0**1·0
set $b/\overline{b}$	0 **0**0·0
raise w	1 **0**0·0
new state	1 **0**0·**I**
lower w	0 **0**0·**I**
raise $b/\overline{b}$	0 **01·0**|**1**

Write 1 operation	
	$(s = 1)$
Description	$w\ b\bar{b}{\cdot}s$
initial state	0 **0**1·1
set $b/\overline{b}$	0 **0**0·1
raise w	1 **0**0·1
new state	1 **0**0·**I**
lower w	0 **0**0·**I**
raise $b/\overline{b}$	0 **01·0**|**1**

During r in state 1, the grounded b line forces the s node to ground (and hence node $\bar{s}$ to V_{dd}), causing the cell's state to change before the $\overline{b}$ line has a chance to fully discharge. Afterwards $\bar{s}$ drives $\overline{b}$ throug

the pass transistor. Since b is already 0, the sense amplifiers will always detect a 0. During w_1, when the w line is asserted, both $b/\overline{b}$ lines are 0, causing both pull-up transistors to conduct which, in turn, results in a metastable state. Once the w line is de-asserted one of the pull-down transistors dominates over the other and the cell eventually reverts back to one of the two quiescent states. Since the state of the cell cannot be predicted after w_1, this operation must be followed immediately by r which will force the cell into a determinate but faulty state.

Analogous behavior is exhibited in the presence of the $\overline{b}$-*sa-0* fault.

***b-sa-1* fault.** The reader can verify that only r and w_0 are affected by this fault. This is because either the b line is always 1 by definition, as in the case of w_1 or as in the case of w_0 in state 0, when the fact that the b line is stuck-at 1 does not alter the cell's state. Table 5 describes the faulty behaviors of an SRAM cell affected by the *b-sa-1* fault.

Table 5: SRAM cell operation with *b-sa-1* fault.

Read operation	
Description	$(s = 0)$ $w\ b\bar{b}{\cdot}s$
initial state	0 11·0
float $b/\overline{b}$	0 $\mathbf{1}\tilde{\mathbf{1}}$·0
raise w	1 $\mathbf{1}\tilde{\mathbf{1}}$·0
b or $\overline{b}$ discharges	1 $\mathbf{1}\tilde{\mathbf{1}}$·0
read $b/\overline{b}$ & lower w	0 $\mathbf{1}\tilde{\mathbf{1}}$·0
raise $b/\overline{b}$	0 11·0
Write 0 operation	
Description	$(s = 1)$ $w\ b\bar{b}{\cdot}s$
initial state	0 11·1
set $b/\overline{b}$	0 **11**·1
raise w	1 **11**·1
new state	1 **11**·**1**
lower w	0 **11**·**1**
raise $b/\overline{b}$	0 11·**1**

An r in state 0 will be misinterpreted by the sense circuitry because both $b/\overline{b}$ lines remain high; therefore, r is not a reliable source of output. When the cell is in state 1, w_0 fails, because b does not force node s to ground and initiate the state transition.

Analogous behavior is exhibited in the presence of the $\overline{b}$-*sa-1* fault.

***w-sa-0* fault.** Since this fault affects the w line, it alters the behavior of all operations. Table 6 describes the faulty behaviors of an SRAM cell in the presence of a *w-sa-0* fault. From this table, we see that r will

Table 6: SRAM cell operation with *w-sa-0* fault.

Read operations		
Description	$(s=0)$ $w\ b\bar{b}{\cdot}s$	$(s=1)$ $w\ b\bar{b}{\cdot}s$
initial state	0 11·0	0 11·1
float $b/\bar{b}$	0 $\tilde{1}\tilde{1}$·0	0 $\tilde{1}\tilde{1}$·1
raise w	**0** $\tilde{1}\tilde{1}$·0	**0** $\tilde{1}\tilde{1}$·1
b or $\bar{b}$ discharges	**0** $\tilde{\mathbf{1}}\tilde{\mathbf{1}}$·0	**0** $\tilde{1}\tilde{\mathbf{1}}$·1
read $b/\bar{b}$ & lower w	0 $\tilde{\mathbf{1}}\tilde{\mathbf{1}}$·0	0 $\tilde{1}\tilde{\mathbf{1}}$·1
raise $b/\bar{b}$	0 11·0	0 11·1

Write operations $(s=0)$		
Description	w_1 $w\ b\bar{b}{\cdot}s$	
initial state	0 11·0	
set $b/\bar{b}$	0 10·0	
raise w	**0** 10·0	
new state	**0** 10·**0**	
lower w	0 10·**0**	
raise $b/\bar{b}$	0 11·**0**	

Write operations $(s=1)$		
Description	w_0 $w\ b\bar{b}{\cdot}s$	
initial state	0 11·1	
set $b/\bar{b}$	0 01·1	
raise w	**0** 01·1	
new state	**0** 01·**1**	
lower w	0 01·**1**	
raise $b/\bar{b}$	0 11·**1**	

fail, since neither $b/\bar{b}$ line is allowed to discharge. Lack of differential voltages on the $b/\bar{b}$ lines yields an unpredictable response of the sense circuitry. Again, r is not a reliable source of output. Since the w line is never asserted, w_1 from state 0 and w_0 from state 1 are also affected, as they do not result in a transition to the desired state. Neither w_0 from state 0 nor w_1 from state 1 are affected, as they effectively "do nothing".

***w-sa-1* fault.** As the reader can verify using Table 1, in the presence of this fault, despite minor differences in the sequence-of-events model, all operations are correct in the FSM model of a single cell.

B SRAM Test Analysis

Here we present the complete analyses of the MATS++, MARCH Y and MARCH C- tests. The reader is reminded that these analyses were performed under the single-fault assumption.

B.1 Evaluation of the MATS++ test

The MATS++ test [3] for a n-word by 1-bit SRAM, is of length $6n$, and is presented below:

$$\text{MATS++} = (w_0^i)^{\updownarrow}(r^i w_1^i)^{n\downarrow 1}(r^i w_0^i r^i)^{1\uparrow n}.$$

First, all the words are initialized to 0. Then, in the fault-free SRAM, each w_1^i in the second march element is preceeded by an r^i which should produce a 0. This march element is performed in the direction from n to 1. In the third march element the first r^i should produce a 1. It is followed by a w_0^i and a second r^i producing a 0. The third march element is performed in the direction from 1 to n.

B.1.1 Evaluation of the MATS++ test for a single cell

Each cell in this test is subject to the following sequence of operations: $w_0 r w_1 r w_0 r$. Table 7 provides the comparison of the fault-free response with the faulty responses of all independently testable faults. The deviations from the fault-free response, as well as the elementary tests within MATS++ are indicated in boldface. From this table it is clear that all independently

Table 7: Evaluation of the MATS++ test for a single cell.

Fault	MATS++ $w_0 r w_1 r w_0 r$	Elementary tests for input stuck-at faults within MATS++
fault-free cell	$-\ 0\ -\ 1\ -\ 0$	
b-sa-0	$-\ 0\ -\ \mathbf{0}\ -\ 0$	$w_0 r \mathbf{w_1 r} w_0 r$
b-sa-1	$-\ \mathbf{1}\ -\ 1\ -\ \mathbf{1}$	$w_0 r \mathbf{w_1} r \mathbf{w_0 r}$
$\bar{b}$*-sa-0*	$-\ \mathbf{1}\ -\ 1\ -\ \mathbf{1}$	$\mathbf{w_0 r} w_1 r \mathbf{w_0 r}$
$\bar{b}$*-sa-1*	$-\ 0\ -\ \mathbf{0}\ -\ 0$	$\mathbf{w_0} r \mathbf{w_1 r} w_0 r$

testable faults will be detected reliably. The reader can verify that for an inverted MATS++ test where each cell undergoes the complementary sequence of operations $w_1 r w_0 r w_1 r$, all independently testable faults will also be detected.

B.1.2 Evaluation of the MATS++ test for n-word by 1-bit SRAM

It is easy to verify that MATS++ detects w^i-*sa-1* faults by noting that MATS++ is an extension of MATS+. Since MATS+ has been shown to detect w^i-*sa-1* faults, and since we can insert any number of 'read' operations, as they do not corrupt the contents of the cells, we conclude that MATS++ also detects these faults.

Given that the w^i-*sa-0* faults are not reliably detectable[5], we conclude that in a n-word by 1-bit SRAM this test can reliably detect $\frac{n+4}{2n+4} \cdot 100\%$ possible input stuck-at faults, which is roughly 50% of faults (for large n) in the input stuck-at model.

B.1.3 Evaluation of the MATS++ test for n-word by l-bit SRAM

The MATS++ test can be extended for a n-word by l-bit SRAM in the following manner:

$$\text{MATS++} = (w^i_{0\ldots0})^{\updownarrow}(r^i w^i_{1\ldots1})^{n\downarrow 1}(r^i w^i_{0\ldots0} r^i)^{1\uparrow n},$$

where $w_{0\ldots0}$ denotes the writing of an all-0 data background, etc. Since, this extended test detects the same faults per cell, as the test for a single cell and that changing the data background does not affect fault coverage.

From the above analyses we conclude that a word-oriented extension of the MATS++ test will reliably detect all the detectable input stuck-at faults, which constitute $\frac{n+4l}{2n+4l} \cdot 100\%$ of all the faults in the fault model; thus for large n the fault coverage is roughly 50%.

B.2 Evaluation of the MARCH Y test

The MARCH Y test [3] for a n-word by 1-bit SRAM, is of length $8n$, and is presented below:

$$\text{MARCH Y} = (w^i_0)^{\updownarrow}(r^i w^i_1 r^i)^{n\downarrow 1}(r^i w^i_0 r^i)^{1\uparrow n}(r^i)^{\updownarrow}.$$

First, all the words are initialized to 0. Then, in the fault-free SRAM, each w^i_1 in the second march element is preceeded and succeeded by an r^i where the former should produce a 0 and the latter a 1. This march element is performed in the direction from n to 1. The third march element is similar, with 0 and 1 interchanged and the march element direction reversed. In the last march element each r^i should produce a 0.

[5] See Section 3.1.1

B.2.1 Evaluation of the MARCH Y test for a single cell

Each cell in this test is subject to the following sequence of operations: $w_0 r w_1 r r w_0 r r$. Table 8 provides the comparison of the fault-free response with the faulty responses of all independently testable faults. As before, the deviations from the fault-free response, as well as the elementary tests within MATS++ are indicated in boldface. From this table it is clear that all

Table 8: Evaluation of the MARCH Y test for a single cell.

Fault	MARCH Y $w_0 r w_1 r r w_0 r r$	Elementary tests for input stuck-at faults within MARCH Y
fault-free cell	$-\ 0\ -\ 11\ -\ 00$	
b-sa-0	$-\ 0\ -\ \mathbf{00}\ -\ 00$	$w_0 r \mathbf{w_1 r} r w_0 r r$
b-sa-1	$-\ \mathbf{1}\ -\ 11\ -\ \mathbf{11}$	$w_0 r \mathbf{w_1} r r \mathbf{w_0 r} r$
$\bar{b}$*-sa-0*	$-\ \mathbf{1}\ -\ 11\ -\ \mathbf{11}$	$\mathbf{w_0 r} w_1 r \mathbf{w_0 r} r$
$\bar{b}$*-sa-1*	$-\ 0\ -\ \mathbf{00}\ -\ 00$	$\mathbf{w_0} r \mathbf{w_1 r} r w_0 r r$

independently testable faults will be detected reliably. It is worth noting that for our fault model MARCH Y is a redundant test, as the additional 'read' operations (in comparison with MATS++) do not increase the fault coverage and their removal would not decrease the existing coverage. The reader also can verify that for an inverted MARCH Y test where each cell undergoes the complementary sequence of operations $w_1 r w_0 r r w_1 r r$, all independently testable faults will also be detected.

B.2.2 Evaluation of the MARCH Y test for n-word by 1-bit SRAM

It is easy to verify that MARCH Y detects w^i*-sa-1* faults by noting that MARCH Y is an extension of MATS+. Since MATS+ has been shown to detect w^i*-sa-1* faults, and since we can insert any number of 'read' operations, as they do not corrupt the contents of the cells, we conclude that MARCH Y also detects these faults.

Given that w^i*-sa-0* faults are not reliably detectable, we conclude that in a n-word by 1-bit SRAM this test can reliably detect $\frac{n+4}{2n+4} \cdot 100\%$ possible input stuck-at faults.

B.2.3 Evaluation of the MARCH Y test for n-word by l-bit SRAM

The combination of the two extensions yield a possible MARCH Y test for a n-word by l-bit SRAM is presented below:

$$\text{MARCH Y} = (w^i_{0\ldots0})^{\updownarrow}(r^i w^i_{1\ldots1} r^i)^{n\downarrow 1}(r^i w^i_{0\ldots0} r^i)^{1\uparrow n}(r^i)^{\updownarrow},$$

where $w_{0\ldots0}$ denotes the writing of an all-0 data background, etc. As before, this extended test detects the same faults per cell, as the test for a single cell and that changing the data background does not affect fault coverage.

The results of the above analyses indicate that a word-oriented extension of the MARCH Y test will reliably detect all the detectable input stuck-at faults, which constitute $\frac{n+4l}{2n+4l} \cdot 100\%$ of all the faults in the fault model. This coverage is identical to that of the MATS++, yet the MARCH Y is longer, as it contains redundant elements (for our fault model).

B.3 Evaluation of the MARCH C- test

The MARCH C- test [3] for a n-word by 1-bit SRAM, is of length $10n$, and is presented below:

$$\text{MARCH C-} = (w^i_0)^{\updownarrow}(r^i w^i_1)^{n\downarrow 1}(r^i w^i_0)^{n\downarrow 1}(r^i w^i_1)^{1\uparrow n}(r^i w^i_0)^{1\uparrow n}(r^i)^{\updownarrow}.$$

First, all the words are initialized to 0. Then, in the fault-free SRAM, each w^i_1 in the second march element is preceeded by an r^i where it should produce a 0. This march element is performed in the direction from n to 1. The next three march elements are similar, with 0 and 1 interchanged and/or the march element direction reversed. In the last march element each r^i should produce a 0.

B.3.1 Evaluation of the MARCH C- test for a single cell

Each cell in this test is subject to the following sequence of operations: $w_0 r w_1 r w_0 r w_1 r w_0 r$. Table 9 provides the comparison of the fault-free response with the faulty responses of all independently testable faults. From this table it is clear that all independently testable faults will be detected reliably. It is worth noting that for our fault model MARCH C- is also a redundant test, as every elementary test is repeated at least twice. The reader also can verify that for an inverted MARCH C- test where each cell undergoes the complementary sequence of operations $w_1 r w_0 r w_1 r w_0 r w_1 r$, all independently testable faults will also be detected.

Table 9: Evaluation of the MARCH C- test for a single cell.

Fault	MARCH C- $w_0 r w_1 r w_0 r w_1 r w_0 r$	Elementary tests for input stuck-at faults within MARCH C-
fault-free cell	$-0-1-0-1-0$	
b-sa-0	$-0-\mathbf{0}-0-\mathbf{0}-0$	$w_0 r \mathbf{w_1 r} w_0 r \mathbf{w_1 r} w_0 r$
b-sa-1	$-\mathbf{I}-1-\mathbf{1}-1-\mathbf{1}$	$w_0 r \mathbf{w_1} r \mathbf{w_0 r w_1} r \mathbf{w_0 r}$
$\bar{b}$*-sa-0*	$-\mathbf{1}-1-\mathbf{1}-1-\mathbf{1}$	$\mathbf{w_0 r} w_1 r \mathbf{w_0 r} w_1 r \mathbf{w_0 r}$
$\bar{b}$*-sa-1*	$-0-\mathbf{0}-0-\mathbf{0}-0$	$\mathbf{w_0} r \mathbf{w_1 r w_0} r \mathbf{w_1 r} w_0 r$

B.3.2 Evaluation of the MARCH C- test for n-word by 1-bit SRAM

It is easy to verify that MARCH C- detects w^i*-sa-1* faults by noting that MARCH C- is an extension of MATS+. In fact, the first, second and fifth match element of MARCH C- constitute MATS+. It suffices to show that the removal of march elements three and four from MARCH C- is possible.

After the second march element the memory is filled with 1s. The third march element fills the memory with 0, and the fourth fills it back with 1s again. Since the effects of march elements three and four cancel each other, i.e. the contents of the memory cells after the fourth march element in MARCH C- is in the same as it was after the second march element, march elements three and four can be removed. By removing these two march elements, as well as the sixth one, from MARCH C-, we obtain MATS+. Since MATS+ has been shown to detect w^i*-sa-1* faults we conclude that MARCH C- also detects these faults.

Since MARCH C- test cannot reliably detect w^i*-sa-0* faults, we conclude that in a n-word by 1-bit SRAM, this test can reliably detect $\frac{n+4}{2n+4} \cdot 100\%$ possible input stuck-at faults, which is roughly 50% of faults (for large n) in the input stuck-at model.

B.3.3 Evaluation of the MARCH C- test for n-word by l-bit SRAM

The combination of the two extensions yield a possible MARCH C- test for a n-word by l-bit SRAM is presented below:

$$\text{MARCH C-} = (w^i_{0\ldots0})^{\updownarrow}(r^i w^i_{1\ldots1})^{n\downarrow 1}(r^i w^i_{0\ldots0})^{n\downarrow 1}(r^i w^i_{1\ldots1})^{1\uparrow n}(r^i w^i_{0\ldots0})^{1\uparrow n}(r^i)^{\updownarrow},$$

where $w_{0\ldots0}$ denotes the writing of an all-0 data background, etc. The reader should note that, for input stuck-at faults, this extended test detects the

same faults per cell, as the test for a single cell and that changing the data background does not affect fault coverage.

The results of the above analyses indicate that a word-oriented extension of the MARCH C- test will also reliably detect all the detectable input stuck-at faults, which constitute $\frac{n+4l}{2n+4l} \cdot 100\%$ of all the faults in the fault model. This coverage of input stuck-at faults is identical to that of the MATS++ and MARCH Y, yet the MARCH C- is longer than either of these two tests.