JOURNAL OF ELECTRONIC TESTING: Theory and Applications 16, 389–395, 2000

New March Tests for Multiport RAM Devices*

KANAD CHAKRABORTY
EDA Laboratory, IBM T.J. Watson Research Center, Yorktown Heights, NY 10598
kanadc@us.ibm.com

PINAKI MAZUMDER
Department of EECS, The University of Michigan, Ann Arbor, MI 48109
mazum@eecs.umich.edu

Received September 9, 1999; Revised January 24, 2000

Editor: K.K. Saluja

Abstract. This paper describes three new march tests for multiport memories. A read (or write) port in such a memory consists of an n-bit address register, an n-to-2^n-bit decoder (with column multiplexers for the column addresses) and drivers, and a K-bit data register. This approach gives comprehensive fault coverage for both array and multiport decoder coupling faults. It lends itself to a useful BIST implementation with a modest area overhead that tests these faults and achieves low test application time.

Keywords: multi-port RAM, simplex and duplex coupling faults, concurrent coupling faults

1. Introduction

In this paper, we propose three new march tests, M1, M2 and M3 for testing multiport memories. We also examine BIST approaches for testing memory devices using these new algorithms. The fault models used in this paper consist of stuck-at, simplex, duplex and concurrent coupling faults [1–3]. The various types of simplex, duplex, complex and concurrent coupling faults are: (a) **simplex idempotent coupling fault:** that is, a positive or negative transition in a memory location C_j causes another memory location C_i to be stuck at a certain value; (b) **simplex inversion coupling fault:** a positive or negative transition in a memory location C_j inverts the contents of C_i; (c) **duplex coupling fault:** a pair of transitions on memory locations C_j and C_k that jointly cause the contents of a location C_i to be forced to a certain value; (d) **complex coupling fault:** similar to a duplex coupling fault, but induced by more than two simultaneous write transitions; (e) **concurrent coupling fault:** a transition on location C_j that causes a location C_i to be prevented from simultaneously undergoing a transition from state s to $\bar{s}$; (f) **decoder fault models:** both single and dual-port row decoder and column multiplexer fault models. Single-port fault models include, for example, a column multiplexer simultaneously choosing two different columns in the same subarray. Dual-port fault models include, for example, two row decoders decoding the same row address pattern in two different ways (i.e., accessing different memory locations). The above coupling faults comprise some of the most likely neighborhood pattern-sensitive faults due to various types of leakage currents, capacitive coupling, and electromagnetic noise affecting a small group of cells in a close vicinity. Also, it can be expected that the complex coupling faults of arbitrary order (i.e. involving arbitrarily

*This research was supported by a grant from the National Science Foundation.

many memory locations) would most likely trigger duplex and concurrent coupling faults.

2. Addressing Sequence

We use the following addressing sequence for pair-wise memory access: $\langle 1, 2\rangle, \langle 1, 3\rangle, \ldots, \langle 1, n\rangle, \langle 2, 1\rangle, \langle 2, 3\rangle, \ldots, \langle 2, n\rangle, \ldots, \langle n, 1\rangle, \langle n, 2\rangle, \ldots, \langle n, n-1\rangle$. Therefore, each address pair $\langle i, j\rangle$ is encountered twice in the sequence—as $\langle i, j\rangle$ and as $\langle j, i\rangle$. In other words, we are using a *redundant addressing sequence* of length $n(n-1)$, unlike the algorithms described in [2] or [3], which use a pair-wise addressing sequence of length $\binom{n}{2}$. This is required in order to get rid of duplex fault masking, as described later. In the following algorithms, each $\Uparrow$ or $\Downarrow$ refers to the dual-port addressing sequence described above, with $\Uparrow$ denoting 'forward' and $\Downarrow$ denoting 'reverse' sequence. Each $\uparrow$ refers to the simplex addressing sequence i.e., the sequence $1 \rightarrow 2 \ldots \rightarrow n$ and $\downarrow$ refers to the reverse sequence. The notations used are defined below:

- $W_k(i)$: single-port write operation of value $k \in \{0, 1\}$ at address i;
- W_k: single-port write operation of value $k \in \{0, 1\}$ at current address;
- $W_{kl}(i, j)$: dual-port write operation of values $k, l \in \{0, 1\}$ on address-pair $\langle i, j\rangle$;
- $D_{kl}(i, j)$: dual-port transition write operation of values $k, l \in \{0, 1\}$ on address-pair $\langle i, j\rangle$;
- $R(i)$: single port read operation of the value expected from address i;
- R: read operation of the value(s) expected from current address or address-pair;
- $R(i, j)$: dual-port read operation of the values expected from address-pair $\langle i, j\rangle$.

3. Algorithm M1

Algorithm M1 is described in Fig. 1. It consists of 8 steps, $S1$ through $S8$. Let us describe steps $S1$ and $S2$ to understand how the algorithm works. The remaining steps will follow easily from the description below.

Let us consider a multiport RAM with n memory addresses.

Step *S*1: A single-port write march W_0 initializes the memory with 0. After this, we march the following

$S1$: $\uparrow_i (W_0), \Uparrow_{i,j} (R(i,j), W_1(i), W_1(j), R(i,j), D_{00}(i,j)), \uparrow_i (R)$

$S2$: $\uparrow_i (W_1), \Uparrow_{i,j} (R(i,j), D_{00}(i,j)), \uparrow_i (R)$

$S3$: $\uparrow_i (W_0), \Uparrow_{i,j} (R(i,j), D_{11}(i,j)), \uparrow_i (R)$

$S4$: $\uparrow_i (W_1), \Uparrow_{i,j} (R(i,j), W_0(i), W_0(j), R(i,j), D_{11}(i,j)), \uparrow_i (R)$

$S5$: $\uparrow_i (W_0), \Uparrow_{i,j} (R(i,j), W_0(i), W_1(j), R(i,j), D_{10}(i,j)), \uparrow_i (R)$

$S6$: $\uparrow_i (W_1), \Uparrow_{i,j} (R(i,j), W_0(i), W_1(j), R(i,j), D_{10}(i,j)), \uparrow_i (R)$

$S7$: $\uparrow_i (W_0), \Uparrow_{i,j} (R(i,j), W_1(i), W_0(j), R(i,j), D_{01}(i,j)), \uparrow_i (R)$

$S8$: $\uparrow_i (W_1), \Uparrow_{i,j} (R(i,j), W_1(i), W_0(j), R(i,j), D_{01}(i,j)), \uparrow_i (R)$

Fig. 1. Algorithm M1.

sequence over each address pair $\langle i, j\rangle$ (from the set of all $n(n-1)$ address-pairs):

1. a dual-port read operation (called a read **protection** operation, see Definition 1 below) on $\langle i, j\rangle$; followed by
2. a single-port write operation of value 1 on address i; followed by
3. a single-port write operation of value 1 on address j; followed by
4. a dual-port read operation (another read **protection** operation) on $\langle i, j\rangle$; followed by
5. a dual-port transition write operation on $\langle i, j\rangle$.

Next, we perform a single-port read operation on all the n addresses in the memory. We now perform step $S2$, as described below.

Step *S*2: A single-port write march W_1 initializes the memory with 1. After this, over each address pair $\langle i, j\rangle$ (from the set of all $n(n-1)$ address-pairs), we march the following sequence:

1. a dual-port read protection operation on $\langle i, j\rangle$; followed by
2. a dual-port transition write operation on $\langle i, j\rangle$.

Next, we perform a single-port read operation on all the n addresses in the memory. After this we perform steps $S3$ through $S8$.

4. Fault Coverage of M1 for Array Faults

We assume that M1 belongs to a test suite that begins with tests for simplex and stuck-at faults (i.e. those involving single-port write and read accesses). Hence, an algorithm like Algorithm C or some other simple functional test [4, 5] may be executed before executing M1.

Definition 1. A **read-protected** write operation is a write operation (single or dual-port) that is preceded by a read operation on the same address(es).

For example, the dual-port write operations in M1 are all read protected, whereas some of the single-port writes are not.

Definition 2. For memory locations c_i, c_j and c_k, $T_f[(j^s, k^r), i_p]$, with s, r and p belonging to the set $\{0, 1\}$, denotes the duplex coupling fault in which a simultaneous transition of c_j to s and c_k to r results in forcing c_i to state p.

Definition 3. For memory locations c_i, and c_j, $T_f[(i^s, j^r), i_{\bar{s}}]$, with s and r belonging to the set $\{0, 1\}$, denotes the concurrent coupling fault (of order 2) in which a transition of memory location c_j from $\bar{r}$ to r prevents the simultaneous transition of memory location c_i from $\bar{s}$ to s.

Theorem 1. *If* (a) *the decoders are fault-free*; (b) *dual-port read operations are fault-free*; (c) *inverting duplex coupling faults* [3] *do not exist*; *and* (d) *there exist no simplex and stuck-at faults*; *M1 detects all single and multiple faults composed of concurrent and duplex coupling faults.*

Proof: Note that by assumption, no simplex and stuck-at fault exists when algorithm M1 is executed. Therefore, no simplex write faults can mask dual-port write faults. Also, in algorithm M1, any single-port write operation on an address i occurring *after* a dual-port write operation involving address i is protected by a read operation that verifies the contents of i. Hence, a single-port write cannot mask a faulty value produced by a dual-port write transition in algorithm M1, since the faulty value will be read before the single-port write is executed.

We first prove that M1 detects all single duplex and concurrent coupling faults. Since the decoders are fault-free, we assume that the address(es) accessed by one or more ports is always the *intended* address. Consider the fault $F = T_f[(j^s, k^r), i_p]$ as the *only fault that exists in the system.* We have four cases:

1. $s = r = 0$: The fault F will be sensitized **only** in steps $S1$ and $S2$, and in none of the others, since only these two steps have the D_{00} transition in them. Suppose $p = 1$. Note that the single port write operations $W_1(i)$ and $W_1(j)$ in the march sequence $\Uparrow_{i,j} (R(i, j), W_1(i), W_1(j), R(i, j), D_{00})$ of $S1$ are themselves fault-free (by assumption, since F is the only fault), also they are protected by an immediately preceding dual-port read. Therefore, after the location-pair $\langle j, k\rangle$ has undergone the D_{00} transition, there is no further write (fault-free or faulty) that affects the state of location i before the next read accessing location i, which verifies its contents. This read operation will be either the dual-port read that includes access of location i together with another address, or the single-port read at the end of $S1$ (or $S2$) if $j = n$ and $k = n - 1$. In the first case, i.e. if either $j \neq n$ or $k \neq n - 1$, a dual-port read will definitely access location i *regardless of the relative values of* i, j *and* k, since the duplex march operation visits each pair $\langle i, j\rangle$ twice, once as $\langle i, j\rangle$, and once as $\langle j, i\rangle$.

   Similarly, if $p = 0$, a similar reasoning holds for $S2$. Therefore, if $p = 1$, this fault is detected in $S1$ and if $p = 0$, this fault is detected in $S2$. Moreover, if $i = j$ or $i = k$ (i.e. the fault is concurrent), then also the first dual-port read operation (accessing address i) after the D_{00} transition on $\langle j, k\rangle$ verifies the contents of i.
2. $s = 0$, $r = 1$: Similar reasoning as above with respect to the D_{01} transition happening in steps $S7$ and $S8$.
3. $s = 1$, $r = 0$: Similar reasoning as above with respect to the D_{10} transition happening in steps $S5$ and $S6$.
4. $s = r = 1$: Similar reasoning as above with respect to the D_{11} transition happening in $S3$ and $S4$.

Next, we prove that M1 detects all multiple duplex and concurrent coupling faults. If multiple faults do not mask one another, there is no problem, since they can be treated as single faults and can be detected in one or more of the steps enumerated above (1–4). Therefore, we shall now consider multiple faults that mask one another. Also, to simplify the fault detection problem, if three or more faults mask one another, we assume that

two of those faults will mask each other. This assumption will be invalid in only those rare cases in which leakage currents and capacitive coupling between cells is weak for pairs of cells but strong for triplets or larger groups of cells.

Consider the occurrence of the two faults, affecting locations $j1$, $j2, k1, k2$ and $i : F1 = T_f[(j1^{s1}, k1^{r1}), i_p]$ and $F2 = T_f[(j2^{s2}, k2^{r2}), i_q]$, with $s1$, $r1$, p, $s2$, $r2$, all q all belonging to the set $\{0, 1\}$. Since we assume that the faults mask one another, $p = \bar{q}$. We have two cases:

1. $s1 \neq s2$ or $r1 \neq r2$: Without loss of generality, let $s1 = 0$ and $s2 = 1$. Then $F1$ can only be sensitized in steps $S1$, $S2$, $S7$ or $S8$, whereas $F2$ can only be sensitized in steps $S3$, $S4$, $S5$ or $S6$. Suppose, without loss of generality, that $F1$ is sensitized in step $S1$ and $F2$ in $S3$. Then, before $F1$ has a chance of getting masked by $F2$, the fault effect created by $F1$ will be verified by the next dual-port read operation that accesses location i and another location in $S1$, *with the following exception*:

   (a) the other location is $j1$ (and not $j1 + 1$); (b) $j1 = n$ (i.e. the last address); and (c) $k1 > i$.

   If (a), (b) and (c) are simultaneously true, the location-pair $\langle j1, i\rangle$ $(=\langle n, i\rangle)$ will not be accessed for verification by the dual-port read operation in $S1$ after the fault-causing dual-port write transition has affected the location pair $\langle j1, k1\rangle (=\langle n, k1\rangle)$. To counter this problem, we need the single-port read march $\uparrow_i (R)$ at the end of $S1$ (see Fig. 1) to verify the contents of all the memory locations.

   Note that in all other cases, location i will be accessed by a dual-port read operation after the fault-causing dual write transition (and before any other write operation on i), since we are using a redundant dual-port addressing sequence (i.e., of length $n(n-1)$, instead of $\binom{n}{2}$).
2. $s1 = s2$ and $r1 = r2$: Again, without loss of generality, let $s1 = s2 = 0$ and $r1 = r2 = 1$. The other three cases can be proved along similar lines.

Each of the two faults will be sensitized in step $S7$ or $S8$. If $F1$ is sensitized in $S7$, $F2$ will be sensitized in $S8$, and vice versa. Before $F2$ can mask $F1$ (or vice versa), the fault effect will be verified by the leading read operation in $S7$ (or $S8$), again with the following exception:

(a) the other location is $j1$ (and not $j1 + 1$); (b) $j1 = n$ (i.e. the last address); and (c) $k1 > i$. As before, for this case, we need the single-port read march at the end of $S7$ to verify the memory contents.

The same reasoning applies for concurrent faults as well (i.e. in the case that the 'victim' location i is equal to one of its 'aggressors'), since any subsequent write operation on victim (or aggressor) location is protected by a read operation. This completes the proof. ■

5. March Tests for Decoder Faults

Each port in a multiport memory has its own dedicated row and column decoders (and column multiplexers in case of a word-oriented memory) and data buffers, The simultaneous operation of two ports in test M1 may mask some decoder faults. For a pair of ports accessing the memory locations, coupling faults between the bit-lines or the access transistors to the word-lines may produce peculiar coupling faults. The following sets of conditions [6] need to be tested to verify the proper functionality of the decoders:

- Single-Port Conditions:
  1. Every address pattern must access a unique memory location that is not accessed by any other address. In case of word-oriented and column-multiplexed RAMs, a single address pattern has two fields—one corresponding to a unique row, and the other corresponding to a unique column (in each subarray of the memory). Since an address pattern consists of two fields—one for row address and one for column address, this implies a unique row access and a unique column access in each subarray.
  2. Every memory location should be accessed by a unique address pattern irrespective of which port or ports access it.
- Cross-Port Conditions:
  1. If two ports select two different memory addresses, each port should be able to access the memory locations selected for read or write.
  2. If two ports select the same memory address, both ports should be able to read from the location.

We assume that the decoders do not exhibit any sequential circuit behavior in presence of faults.

$S1$: $\uparrow_i (W_0), \uparrow_i (R(i), W_1(i))$

$S2$: $\uparrow_i (W_1), \uparrow_i (R(i), W_0(i))$

Fig. 2. Algorithm M2.

5.1. *M2: Test for Single-Port Conditions*

To test that every address pattern accesses a unique memory location, and also that any memory location is accessed by a unique address pattern, the test in Fig. 2 can be performed.

Note that if a memory location is *not* accessible by any address from a port, by *is* accessible from another port, then the above test will detect this fault, for the following reason: such a memory location will be stuck-at some constant value, and will fail to be initialized to either 0 or 1. Hence, a port that can access this location will detect the fault during the read operation in either $S1$ or $S2$. If a memory location is completely inaccessible by *any* address from any port, it cannot be detected.

5.2. *M3: Test for Cross-Port Conditions*

To test for cross-port decoder faults between ports P1 and P2, we need to have two different columns, C_{P1} and C_{P2}, to be permanently selected by P1 and P2 during the test. This would sensitize a fault that causes P1 (P2) to access a row $R2$ ($R1$) belonging to P2 (P1), since the memory location at the crosspoint of $R2$ ($R1$) and C_{P2} (C_{P1}) would then be spuriously accessed. Therefore, the test shown in Fig. 3 can be performed.

6. BIST/DFT Techniques for the Above Algorithms

The runtime complexity of M1 and M3 is $O(n^2)$, n being the number of memory locations, which makes them quite expensive. M2 is linear in the number of memory locations.

Since two of these tests have quadratic complexity, for running these tests, we advocate the use of multiple read and write ports in the RAM (instead of using only two ports) for achieving faster testing time. This approach was also prescribed in our earlier paper [2]. This test approach employs a boundary-scan interface and the generic test architecture for this approach is illustrated in [2].

With this approach, we need 2 write ports and 2 read ports (a total of 4 ports) to apply algorithm M1, since the longest dual-port march operation in M1 comprises 2 single-port write operations, 1 dual-port write operation (protected by a dual-port read), and 2 dual-port read operations. Use of 4 ports allows us to have each read (or write) operation follow the previous operation by half a clock cycle, instead of one clock cycle. For example, consider the second march element in sequence $S1$, namely: $\Uparrow_{i,j} (R(i, j), W_1(i), W_1(j), R(i, j), D_{00}(i, j))$. In this scenario:

1. Read-ports 1 and 2 execute the $R(i, j)$ operations on all the $n(n-1)$ pairs in memory.

For all read/write port-pairs (P1,P2):

1. Set up a fixed column address for P1 and a fixed column address for P2.
2. Initialize the memory with 0.
3. Perform the duplex read-write march $\Updownarrow_{i,j} (R(i, j), W_{11})$ cycling through all the rows, but holding the column address pattern fixed to the preset value for each port (C_{P1} for P1 and C_{P2} for P2). P1 cycles through the rows in forward sequence (i.e. from row 0 through row $r-1$, r being the total number of rows) and P2 cycles through the rows in the reverse sequence (from row $r-1$ to 0). For each address-pair $\langle i, j \rangle$, P1 and P2 first perform the dual-port read operation $R(i, j)$ and after one clock cycle, perform the dual-port write operation W_{11}.
4. Repeat above step with '11' replaced by '00'.

Fig. 3. Algorithm M3.

2. Write-port 1 executes $W_1(i)$ with a time-lag of half a clock cycle behind Step 1. It goes through an addressing sequence of length $n(n-1)$ as follows:

$$1 \to 1 \to \cdots (n-1 \text{ times}) \, 1,$$
$$2 \to 2 \to \cdots (n-1 \text{ times}) \, 2,$$
$$\cdots$$
$$n \to n \to \cdots (n-1 \text{ times}) \, n$$

3. Write-port 2 executes $W_1(j)$ with a time-lag of half a clock cycle behind Step 2. It goes through an addressing sequence of length $n(n-1)$ as follows:

$$2 \to 3 \to \cdots n,$$
$$1 \to 3 \to \cdots n,$$
$$\cdots$$
$$1 \to 2 \to \cdots n-1$$

4. Read-ports 1 and 2 execute the $R(i, j)$ operations on all the $n(n-1)$ pairs in memory with a time-lag of half a clock cycle behind Step 3.
5. Write-ports 1 and 2 execute the $D_{00}(i, j)$ dual-port transition half a clock cycle behind Step 4.

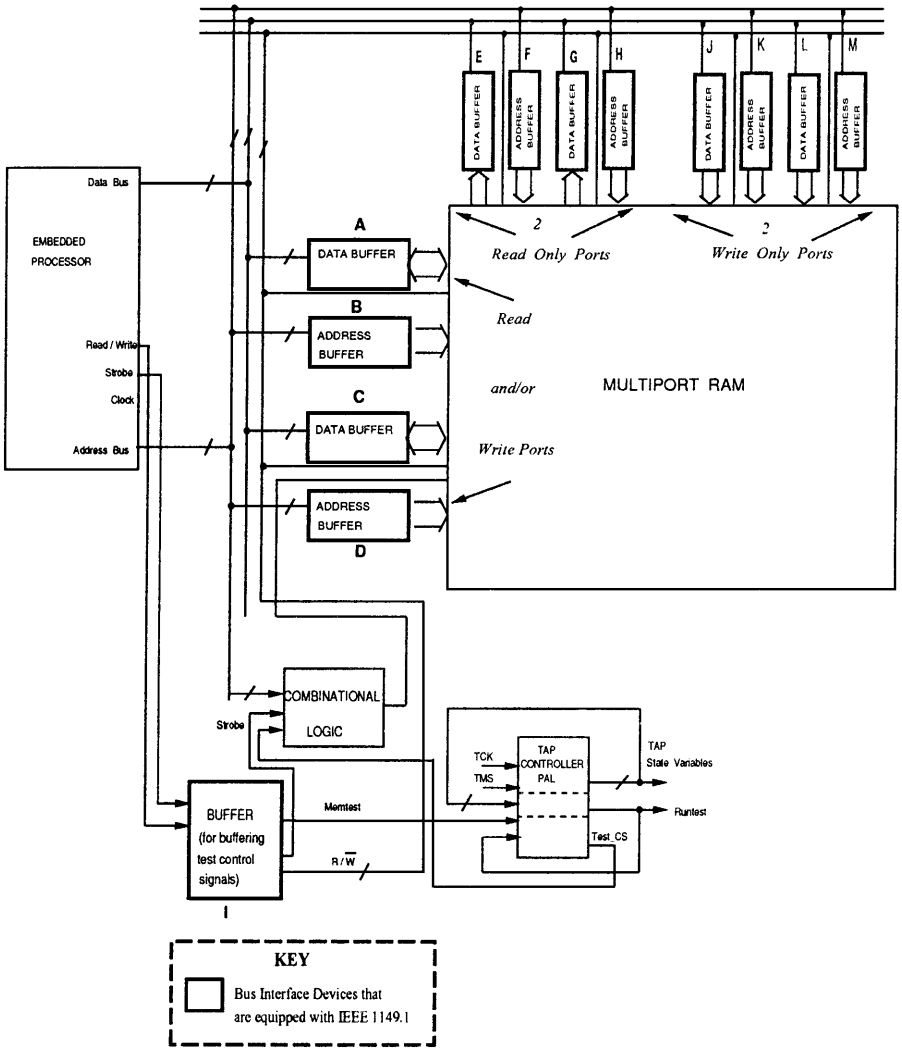

Fig. 4. A multiport embedded RAM; bus interface devices equipped with IEEE 1149.1 have been labeled **A** through **M**).

Each port has a dedicated address buffer, address decoder and data buffer. The address buffer of each port has a register that is reconfigured as a sequence generator (such as a pseudorandom pattern generator (PRPG) or a counter) in BIST mode. The data buffer of each read port is reconfigured as a parallel signature analyzer (PSA). Using IEEE 1149.1 compliant scan-test devices, a schematic architecture of a system incorporating the above hardware is shown in Fig. 4.

The different modes of operation during testing are described in [7]. For executing algorithm M3 using this architecture, we need separate address buffers for row and column addresses at each port of the RAM.

If the required number of read and write ports are already present in the RAM, the hardware overhead is quite modest. If the required number of read and write ports are not present in the RAM, BIST-enhancement comprising of adding extra ports to the RAM array, may be needed. However, given the proliferation of multiport memories (sometimes, with as many as 10–12 ports) for high-speed computing and high-bandwidth communication nowadays, BIST enhancement will most probably not be needed.

A nice feature of this scheme is that the multiple read and write ports (used for applying algorithm M1) can test themselves before applying M1 via algorithm M2 and M3. Hence, we recommend the following test sequence:

1. Algorithm M2 and M3 (which considers all ports singly and pairwise).
2. Algorithm M1.

7. Conclusion

This paper describes new march algorithms and a novel BIST scheme to test all the ports of a multiport RAM comprehensively for single and cross-port faults, and the memory array for duplex and concurrent coupling faults. In this scheme, the ports test themselves (i.e. the decoders) first for single and cross-port faults, using algorithms M2 and M3, and then perform comprehensive array testing using algorithm M1. We have also described the BIST scheme involving the use of IEEE 1149.1-compliant scan test devices to run these tests. A test architecture comprising these scan test devices has been used [2, 3] for testing of single-port static RAMs, and has been found to be very efficient in terms of the test application time.

References

1. V.C. Alves, M. Nicolaidis, P. Lestrat, and B. Courtois, "Built-in Self-Test for Multiport RAM's," *Proc. IEEE International Conference on Computer-Aided Design (ICCAD)*, Santa Clara, USA, Nov. 1991, pp. 248–251.
2. K. Chakraborty and P. Mazumder, "A Programmable Boundary Scan Technique for Board-Level, Parallel Functional Duplex March Testing of Word-Oriented Multiport Static RAM's," *Proc. of the European Design and Test Conference*, Paris, France, March 1997, pp. 330–334.
3. M. Nicolaidis, V.C. Alves, and H. Bederr, "Testing Complex Couplings in Multiport Memories," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, Vol. 3, No. 1, pp. 59–71, March 1995.
4. M. Marinescu, "Simple and Efficient Algorithms for Functional RAM Testing," *Proc. IEEE International Test Conference*, Nov. 1982, pp. 236–239.
5. S.M. Thatte and J.A. Abraham, "Testing of Semiconductor Random-Access Memories," *Proc. 7th Annual International Conference on Fault-Tolerant Computing*, 1977, pp. 81–87.
6. A.A. Amin, M.Y. Osman, R.E. Abdel-Aal, and H. Al-Muhtaseb, "New Fault Models and Efficient BIST Algorithms for Dual-Port Memories," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 16, No. 9, pp. 987–1000, Sept. 1997.
7. Texas Instruments, "Boundary-Scan Logic IEEE Std. 1149.1 (JTAG): 5 V and 3.3 V Bus-Interface and Scan Support Products," *Data Book, Advanced System Logic Products*, Texas Instruments, 1994, pp. A23–A30.