Simulation-Based Test Algorithm Generation and Port Scheduling for Multi-Port Memories

Chi-Feng Wu, Chih-Tsun Huang, Kuo-Liang Cheng, Chih-Wea Wang, and Cheng-Wen Wu
Department of Electrical Engineering
National Tsing Hua University
Hsinchu, Taiwan
cww@ee.nthu.edu.tw

ABSTRACT

The paper presents a simulation-based test algorithm generation and test scheduling methodology for multi-port memories. The purpose is to minimize the testing time while keeping the test algorithm in a simple and regular format for easy test generation, fault diagnosis, and built-in self-test (BIST) circuit implementation. Conventional functional fault models are used to generate tests covering most defects. In addition, multi-port specific defects are covered using structural fault models. Port-scheduling is introduced to take advantage of the inherent parallelism among different ports. Experimental results for commonly used multi-port memories, including dual-port, four-port, and *n*-read-1-write memories, have been obtained, showing that efficient test algorithms can be generated and scheduled to meet different test bandwidth constraints. Moreover, memories with more ports benefit more with respect to testing time.

1. INTRODUCTION

Multi-port memories are commonly used components in VLSI systems, such as register files in microprocessors, storage for media or network applications. The content of a multi-port memory can be accessed through different ports simultaneously. This feature is especially valuable for high speed processors, media processors, and communication processors. However, multi-port memories require more testing effort since all ports have to be verified.

A multi-port memory is usually tested as several single-port memories, which are tested separately with test algorithms developed for single port memories. This approach is simple and easy to deploy, but there are two major problems with it.

The first is the lack of detection for multi-port specific defects that may occur in the memory. Multi-port memories need to be verified with more fault models related to inter-port defects. That makes the complexity of multi-port fault models significantly higher than the complexity of single-port fault models.

The second is the inefficiency of the test procedure, because the inherent parallelism of multi-port memories is not fully utilized during testing. Test scheduling for a single-port memory is trivial, i.e., the entire test algorithm is applied to the one and only port. When there are multiple ports, the test algorithm defines tests for all ports. Cell faults, such as stuck-at faults and transition faults, are testable through each one of the ports. On the other hand, an inter-port fault is more difficult to test than a cell fault because the fault activation needs not only a specific address, operation, and data pattern, but also a specific port configuration. Test scheduling is important when testing multi-port memories because testing time can be significantly reduced by proper *test scheduling*.

Functional fault models and tests for two-port memories have been proposed in [1, 2], The complexity of the tests is $O(N^m)$, where N is the size of the memory, and m is the number of ports [1]. When the address scrambling scheme is known, the testing time can be reduced to $O(N)$, but with a large constant factor, e.g., ranging from $81N$ to $269N$ for two-port memories [2]. Structural fault models, such as the adjacent bit-line short, has been used in [3–5]. In general, functional fault models covers more defects, but require longer testing time than structural fault models. Structural fault models are based on the circuit structure of the memory, therefore deriving tests for them need more design information. Also, test patterns for structural fault models are usually not as regular as test patterns for functional fault models.

In this paper, we present a methodology that adopts both functional fault models and structural fault models. The purpose of our methodology is to minimize the testing time while keeping the test algorithm in a simple and regular format of March test for easy test generation/scheduling, fault diagnosis, and built-in self-test (BIST) circuit implementation [6–8]. Several popular RAM fault models, such as stuck-at faults, coupling faults, etc., are used to cover most memory defects, while structural fault models are used to cover defects specific to multi-port memories. Port scheduling is introduced to further optimize the testing time. All test algorithms presented in this paper have been verified with the the fault simulator that we have developed and constantly improved, called RAMSES [9].

2. FAULT MODELS AND DEFINITIONS

Fault models are defined to cover defects in memory cell arrays, address decoders, and read/write circuits. Functional fault models and March algorithms are widely used in the industry for this purpose [10]. Several popular fault models are introduced to demonstrate our methodology, including stuck-at fault (SAF), transition fault (TF), address decoder fault (AF), stuck-open fault (SOF), read disturbance fault (RDF), and coupling fault (CF), for which we consider state coupling (CFst), inversion coupling (CFin), and idempotent coupling (CFid). Table 1 shows several popular March algorithms, following the notation of [10]. March tests are normally short and easy to generate.

DAC 2001, June 18-22, 2001, Las Vegas, Nevada, USA.

Table 1: Some March algorithms.

Name	Algorithm
MATS++	$\Updownarrow(w0);\Uparrow(r0,w1);\Downarrow(r1,w0,r0)$
March X	$\Updownarrow(w0);\Uparrow(r0,w1);\Downarrow(r1,w0);\Updownarrow(r0)$
March Y	$\Updownarrow(w0);\Uparrow(r0,w1,r1);\Downarrow(r1,w0,r0);\Updownarrow(r0)$
March C−	$\Updownarrow(w0);\Uparrow(r0,w1);\Uparrow(r1,w0);\Downarrow(r0,w1);$ $\Downarrow(r1,w0);\Updownarrow(r0)$

Structural fault models specific to multi-port memories are inter-port bridge or short faults, including inter-port *word line shorts* and inter-port *bit line shorts*. Figure 1 shows a two-port memory topology example, in which defects result in a word line short and a bit line short. The bit lines are simplified as single-ended for easy reading, but we actually consider differential pairs and all combinations of shorts during fault simulation.

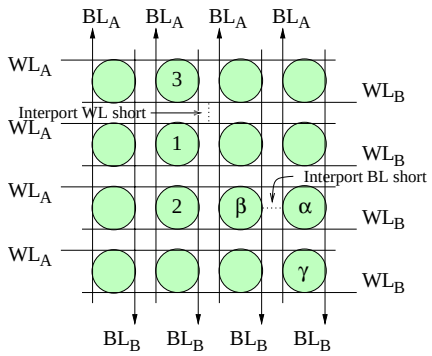

Figure 1: Example of two-port memory layout topology.

When an inter-port word line short occurs as in Figure 1, one of the possible results is shown in Figure 2. Address 2 of port B has a multiple access to Cell 2 and Cell 3 when port A is accessing Cell 1. The resulting value of a read to multiple cells depends on the memory design: possible faulty results are the logic-and or logic-or of the two cells. When an inter-port bit line short occurs as in Figure 1, one of all possible results is shown in Figure 3. Port A Address α has a multiple access to Cell α and Cell β, so has port B Address β. The resulting value of a read to multiple cells depends on the memory design: possible faulty results are the logic-and or logic-or of the two cells. Note that Figure. 2 and 3 describe possible results because an inter-port short can lead to more than one faults on the same bit line or word line.

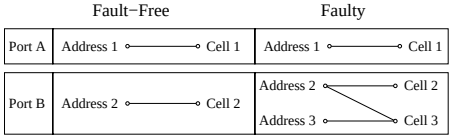

Figure 2: Functional behavior of an inter-port word line short.

Availability of the physical information determines the complexity of inter-port faults. When the address scrambling scheme is unknown, all possible ways of short between all addresses have to be considered. From Figure. 2 and 3, when the physical information is missing, the complexity of inter-port word and bit line shorts are $O(N^3)$ and $O(N^2)$, respectively, if they are mapped to functional faults, i.e., address decoder faults.

On the other hand, given the address scrambling data, test algorithms can be developed to detect most likely shorts in the circuit. The complexity is reduced to the order of the number of bit lines and word lines, i.e., $O(N^{\frac{1}{2}})$ to $O(N)$, depending on the aspect ratio of the memory layout.

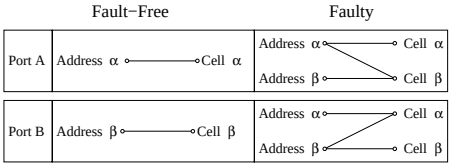

Figure 3: Functional behavior of an inter-port bit line short.

3. TEST STRATEGY

A multi-port memory consists of multiple ports that access the same cell array with their own read/write circuits and address decoders. Applying a March test (such as March C−) on one port can detect most defects in the memory, including the cell array, the port's read/write circuit and the port's address decoder.

Our test strategy is to test the cell array, read/write circuits and address decoders using functional fault models that have been well-developed for single port memories. Inter-port specific defects are covered using structural fault models. The specification of logical address fields such as word line select, bit-line select, I/O select, etc., are required in order to derive the address translation. An example of address translation is depicted in Figure 4.

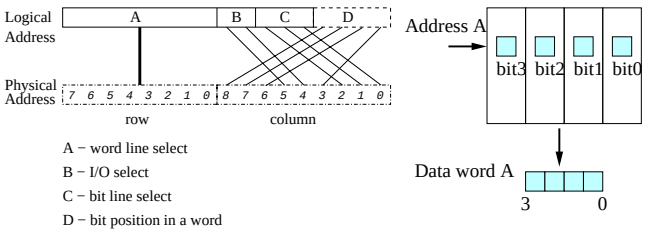

Figure 4: Scrambling.

We propose a novel test pattern that extends March algorithms to test structural fault models that involving neighboring cells on physical locations. The neighboring cells, called *March guards*, are shown in Figure 5. For each address, four reads are defined on its neighboring cells, denoted as r^{N}, r^{S}, r^{E}, and r^{W}, respectively. When reaching the boundary, a March guard can be either degraded to a no-operation (NOP) or pushed-back to the base (B) cell, depending on the implementation preference. The NOP can be implemented as a *dummy read*, i.e., a read without comparison with the fault free value. When degraded to a NOP, the detection capability is slightly reduced but can be recovered by proper port-scheduling, which is explained in the next section.

The expected value of a March guard depends on the address sequence and the data background, as shown in Figure 5. When the address sequence is ascending and the march element changes the base cell from 0 to 1, the expected value for r^{N} and r^{S} are 0 and 1, respectively. One or more of the four reads can be used in a March algorithm to detect memory faults. Because of the regularity of March tests, the expected value in $(r^{\mathrm{N}}, r^{\mathrm{S}})$ or $(r^{\mathrm{W}}, r^{\mathrm{E}})$ are always complementary except on the boundary, therefore they provide a good and simple mechanism for checking line shorts for both AND-type and OR-type shorts.

March guards is an extension to conventional March tests. A

March test with guards for a two-port memory is illustrated in Figure 6. The test algorithm is executed as a normal MATS++ [10] for port A. Two test elements of March guards, i.e.,$(r^{\mathrm{N}}1, r^{\mathrm{S}}0)$ and $(r^{\mathrm{W}}1, r^{\mathrm{E}}0)$, are applied to port B. For proper scheduling of the test elements, a NOP (shown as "$-$") is used to match the timing. In M_1, when port A executes $(r0, w1)$, port B executes $(r^{\mathrm{N}}1, r^{\mathrm{S}}0)$ using the same address sequence, i.e., port B is checking the adjacent cells. When March guards are employed, $(r^{\mathrm{W}}1, r^{\mathrm{E}}0)$ detects word line shorts, while $(r^{\mathrm{N}}1, r^{\mathrm{S}}0)$ can detect word line shorts and bit line shorts. In Figure 1, for example, a read from α or β can detect the bit line short, a read from γ can also detect the bit line short, and a read from 2 can detect the word line short. In general, r^{N} and r^{S} are more versatile, but r^{W} and r^{E} are still useful for certain special conditions, such as boundaries. Their applications will be demonstrated later.

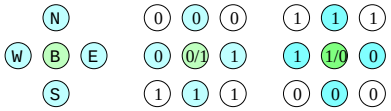

Figure 5: March guards.

	M_0	M_1	M_2
Port A	$\Updownarrow (w0)$	$\Uparrow (r0, w1)$	$\Downarrow (r1, w0, r0)$
Port B	$(-)$	$(r^{\mathrm{N}}1, r^{\mathrm{S}}0)$	$(r^{\mathrm{W}}1, r^{\mathrm{E}}0, -)$

Figure 6: March X with March guards extension.

4. TEST ALGORITHM GENERATION

To generate test algorithms efficiently for various architectures of multi-port memories, a systematic methodology is proposed based on TAGS, a simulation-based test algorithm generator for RAM [6]. Test algorithm generation with port scheduling is an extension to TAGS, named TAGS-PS. On fault simulation for multi-port memories, several assumptions are made:

1. Each memory port has a preset access priority. When accessing an address simultaneously from more than one ports, the operations are arranged according to the order of access priorities, from high to low. For simplicity, we assign the port with the highest access priority as port 1, the next as port 2, and so on.
2. Simultaneous writes to an address from multiple ports are considered as invalid operations.
3. For *n*-read-1-write memories, its write port and one of the read ports are combined into a read/write pair during the operation of cell fault detection. The write port is assigned port 1, without loss of generality.

The test generation methodology provides automatic test generation by minimizing the testing time and guarantees the fault coverage. The test generation algorithm consists of three phases. The procedure is stated as follows. An example will be given later for dual-port memories.

1. Base algorithm generation
   (a) Generate a single port complete test, i.e., a 100% fault coverage test for all cell array faults. The test is performed by accessing port 1, initially.
   (b) Append additional tests for multiple-port specific faults, i.e., address decoder faults (AFs) for each port and inter-port faults. There are possibilities that these tests require simultaneous operations on two or more read/write ports.

Consequently, the phase consists of three sections: the section of a single-port complete test, the section of AFs, and the section of inter-port faults, as shown in Figure 7(a). To detect multi-port specific faults, one or more March guards are inserted, therefore two or more ports may be accessed simultaneously in the test sections 2 and 3, as indicated in the figure.

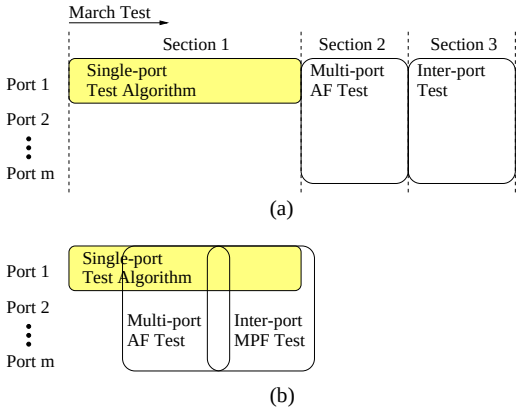

Figure 7: (a) The complete test (b) Port scheduling and test reduction.

2. Port scheduling

   Each test element in the base algorithm has certain degree of freedom, i.e., there may be other elements that can achieve the same fault detection capability. Port scheduling is to search possible compaction ways to reduce the testing time. The procedure is as follows.

   (a) Initial test set contains one or more base algorithms.
   (b) For each test in the test set, select one that contains all test elements in test sections 2 and 3.
   (c) Search for appropriate positions where the selected element matches an element in test section 1. When the selected element is on port i and $i \neq 1$, the matched element in section 1 can be swapped with the test element in port i only when it is a NOP. A *compact* test is generated by embedding the selected test element in test section 1 and deleting it from its original position. All possible compact tests are generated and appended to the test set.
   (d) Simulate each and every test in the test set with RAMSES [9], then delete incomplete tests. Complete tests are sorted by their test length. A threshold value is set for keeping additional tests that are not the shortest test. The default threshold value is 0, i.e., only the tests with the shortest test length are kept in the test set, others are deleted.
   (e) Repeat steps 2b to 2d using the new test set until no further compaction is possible. Report the shortest test in the test set as the final test.

Test section 1 of the base algorithm consists of test elements for port 1. There are certain degree of freedom for test elements in other test sections to be embedded in section 1 since all ports except port 1 perform NOPs in section 1. Moreover, the test elements in port 1 can be swapped with other ports with NOPs, which increases the degree of freedom for further compaction. As indicated in Figure 7(b), the test algorithm is compacted after the scheduling because the detection capability of test elements in various test sections are overlapped. The port scheduling steps will be illustrated in the following example.

3. Redundancy check

   (a) Examine the redundancy among the test for March operations which perform read for only one port and perform NOP for all other ports, called *dangling reads*. Search for *dangling reads* and verify their redundancy by RAMSES. Delete redundant operations.

   (b) Repeat until no further redundancy can be found.

   The detection capability of a dangling read is likely to be covered after port scheduling, e.g., the read operation for detecting SOF through port 1 can be removed because of the insertion of March guards.

For a common dual-port memory, which has two separate read/write ports, the test generation is illustrated in Figure 8. Firstly, a 12N test is generated to cover all SAF, TF, SOF, RDF, and CFs (see test section 1 of Figure 8(a)). A test for AFs for each read/write port is generated as test section 2, and an inter-port specific test is generated as test section 3. After the base algorithm is determined, the port-scheduling is performed.

- Read/write operations in test section 1 are not restricted to the specific port. The operation can therefore be swapped between different ports with the same fault coverage. The port swapping provides the degree of freedom for the compaction of multiple-port operations. For example, test section 3 can be directly matched and embedded in test section 1, as shown in Figure 8(b). Test section 2 can only be embedded into test section 1 after the port swapping of the two consecutive *rwr* operation, resulting in the algorithm in Figure 8(c). Different orders of compaction steps will alter the result. In general, dealing with the test elements of inter-port faults first obtains better result than dealing with the test elements of AFs first, because the constraints of testing AFs are less strict and can be dealt with after the compaction of other tests. However, all possible ordering can be applied to explore the search space for better results. Shorter tests are considered as better ones.

- Moreover, some tests such as that for inter-port faults have alternatives. Figure 9(a) shows two equivalent patterns to detect the inter-port shorts between port i and port j. The patterns also have varieties for different access directions or different data backgrounds. The variations increase the degree of freedom to generate the final test. A possible pattern combination for all inter-port shorts is shown in Figure 9(b). The result of port scheduling helps derive alternatives depending on the choice of equivalent tests. Again, an optimized test is selected among different alternatives, according to the simulation results of RAMSES and the test lengths.

After port scheduling, some test elements may still have redundancy because cell array faults may be covered by March guards. The dangling reads are examined for redundancy because removing the operation and other NOPs will not affect correct timing. As in Figure 8(c) and (d), three March operations are tested and two of them are stripped. Consequently, a 10N algorithm is generated for a dual-port memory.

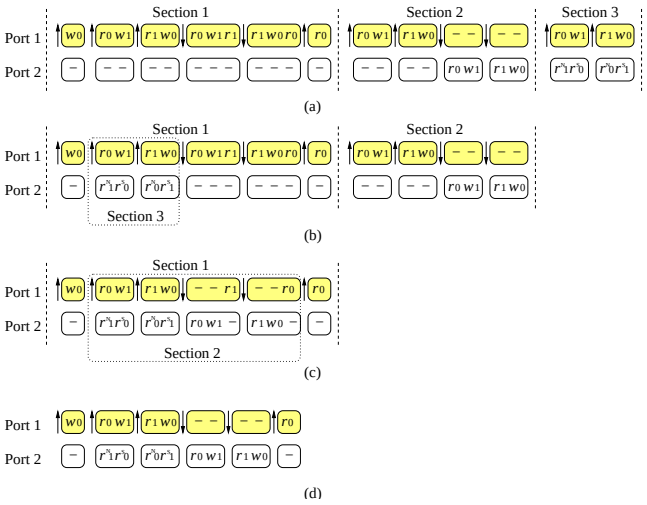

Figure 8: (a) The complete test; (b) Port scheduling for test section 3; (c) Port scheduling for test section 2; (d) Redundancy reduction.

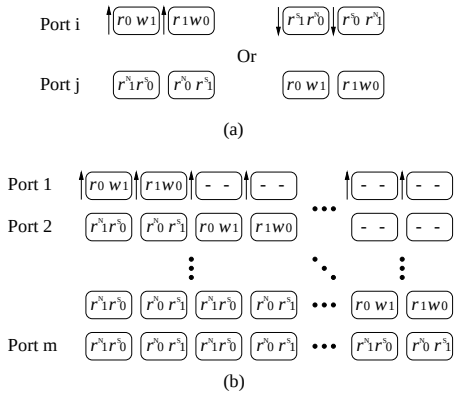

Figure 9: Test of inter-port faults for multi-port memories.

The base algorithm generation and the port scheduling are simple and systematic. In the base algorithm generation, the target fault models are cell array faults. Multi-port address decoder faults and inter-port specific faults are detected in different test sections. The separation of the target faults significantly reduces the complexity of automatic test generation. Once the test is completed, TAGS-PS can be used to compact the algorithm. The systematic methodology of port scheduling prevents the test generation from complicated manual derivations. When including new inter-port fault models, only the new test constraints are required for generating new test sections. RAMSES and TAGS are both easily extensible for new fault models.

5. EXPERIMENTAL RESULTS

We demonstrate the test generation and port scheduling results with several commonly used multi-port memories, including dual-port, four-port, and n-read-1-write memories.

5.1 Dual-port memories

The test generation for dual-port memories is already discussed in the previous section. The final test is shown in Figure 8(d). The test length is $10N$.

5.2 Four-port memories

The test generation methodology for a dual-port memory can be generalized for other multi-port memories, assuming each port is a read/write port. After port scheduling, the overall test algorithm is shown in Figure 10 and the scheduling result is highlighted. Note that the trailing read element is removed as opposed to the dual-port case, because the element becomes redundant after the additional *rw* elements for address decoder faults are inserted.

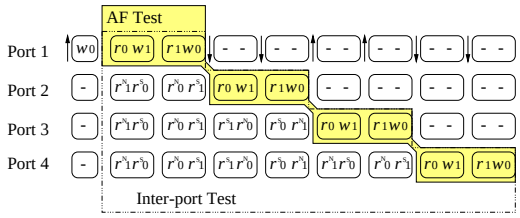

Figure 10: 100% test for four read/write-port memories.

As the number of read/write ports increases, the testing time is dominated by the test for multi-port AFs. The complexity of the March test for multi-port memory is

$$T(N) = \begin{cases} (4m+2)N & \text{for } m = 2, \\ (4m+1)N & \text{for } m > 2, \end{cases} \quad (1)$$

where m is the number of ports.

5.3 n-read-1-write memories

For a memory of one write port and several separate read ports, the test elements of multi-port specific faults are different from that for memories of several read/write ports. The write operation can only be applied to the write-only port.

Figure 11 shows two test categories to detect AFs in n-read-1-write memories. The use of r^{W} and r^{E} are restricted. They can be used only when March guards are pushed back to the base cell on the boundaries.

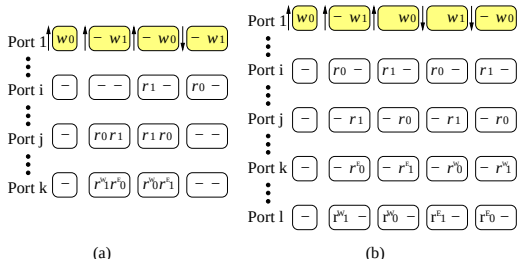

Figure 11: Test alternatives of address decoder faults for n-read-1-write memories.

The write operation has to be applied through the write port (port 1), as shown in Figure 11(a). To detect the AFs of port 1, tests of both port 1 and port i are required. For AFs of read ports, the combination test of port 1 and port j, or port 1 and port k is needed. Figure 11(b) shows four alternative test elements for AF. The test elements of port 1 and port i detects the AFs for port 1 and i. The test elements of other read ports, together with the test elements of port 1, detects the AFs of port j, k and l. All the three patterns are equivalent for the AFs of the read-only ports.

Similarly, Figure 12(a) illustrates the pattern for the detection of inter-port faults between each read port and the write-only port. For the inter-port faults between two read ports, the pattern in Figure 12(b) can be applied, with proper write operation of port 1.

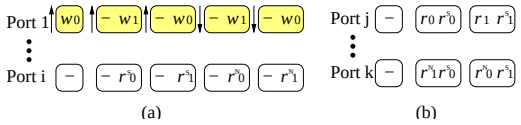

Figure 12: Test alternatives of inter-port faults for the n-read-1-write memories.

Results for 2-read-1-write and 6-read-1-write memories are shown in Figures 13 and 14, respectively. The time complexity can be summarized as

$$T(N) = \begin{cases} 13N & \text{for } 1 < n \leq 4, \\ 15N & \text{for } 4 < n \leq 8, \\ (1+4\times(\lfloor \log n \rfloor + 1))N & \text{for } n > 8, \end{cases} \quad (2)$$

where $n = m - 1$ and m is the total number of ports.

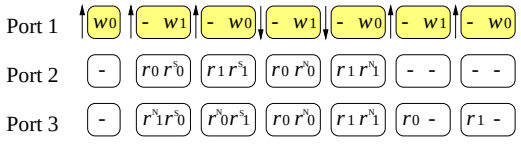

Figure 13: 100% test for the 2-read-1-write memories.

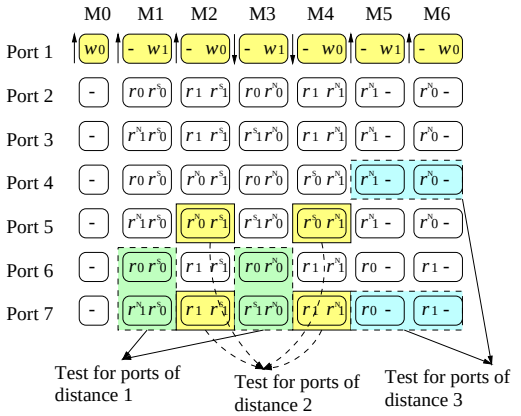

Figure 14: Port scheduling of inter-port faults for the n-read-1-write memories.

6. DISCUSSION

For simplicity, the above cases have been discussed for bit-oriented memories. When test generation is applied to a word-oriented memory, the test algorithm consists of multiple data backgrounds. The single-port algorithm is a *cocktail March test* [6]. The test insertion for address decoder faults and inter-port faults are similar to the procedure for bit-oriented memories. Note that for word-oriented memories, in addition to address scrambling, the data may also be scrambled, as illustrated in Figure 4. In this case,

the March guards are applied in parallel, therefore the test is more efficient than testing bit-oriented memories.

Testing time comparison for several commonly used multi-port memories are listed in Table 2. The testing time are linear with small constant factors. Thanks to the efficiency of port scheduling, the testing time is shorter than previous works, especially for large number of ports. Certain hard-to-detect cell faults that need specific test elements to detect, e.g., stuck-open or read disturbance, are easier to detect with port scheduling. For example, to test SOF in a single-port memories, we need $\Uparrow(r0,w1,r1)$ or $\Uparrow(r0,w1,r1)$ in the test algorithm. With port scheduling and March guards, these specific test elements are no longer needed, since fault effects in the cell array are observable through other ports.

Our methodology has a requirement that the address scrambling of the memory core must be available. Functional tests do not have this limitation because the scrambling data is only used to lower the complexity, i.e., from $O(N^m)$ to $O(N)$, though it is not mandatory. However, even with the scrambling data, the testing time is still very long, e.g., most of them are longer than $100N$ [1,2]. When the long testing time is affordable, complex functional tests may have high fault coverage on complex fault models. The investigation of complex fault models are beyond our discussion here, i.e., we only compare our approach with recent works that adopt structural fault models for multi-port specific defects. Results in Table 2 show that TAGS-PS generates more efficient tests than previous works when there are more ports.

Table 2: Testing time comparison (results in [5] are estimated for the best case).

Memory type	TAGS-PS results	Zhao *et al.* [5]
Dual-port	$10N$	$10N$
m-port, $m>2$	$(4m+1)N$	$10(m-2)N$
n-read-1-write $n>1$	$13N$ for $1<n\leq 4$ $15N$ for $4<n\leq 8$ $(1+4\times(\lfloor\log n\rfloor+1))N$ for $n>8$	$10(n-1)N$

Test bandwidth is a common concern in practice. To apply parallel testing on a multi-port memory, the need for test bandwidth increases proportionally to the port number. When the test pins are not enough to deliver patterns simultaneously to all ports, trade-offs have to be made between testing time and number of test pins by re-scheduling. Table 3 shows the rescheduling results when available test pins can provide parallel access to ports ranging from 2 to 6. In case only one port can be accessed, the test will be degraded to testing each port separately and suffers from fault coverage loss on inter-port faults. In Figure 15, we show a four-port rescheduling with a limited test bandwidth for only 2 ports accessed simultaneously. The testing time is increased from $17N$ to $25N$. Rescheduling can also reduce the bandwidth requirements without testing time penalty in certain cases, e.g., using 4 ports to test a 6-port memory as shown in Table 3. Proper multiplexing or dispatching has to be arranged for delivering test patterns.

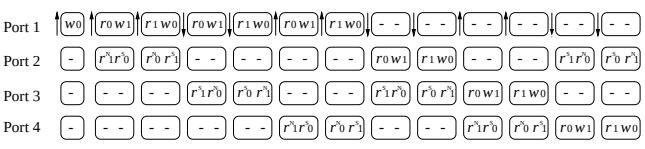

Figure 15: A four-port test rescheduling example.

Built-in self-test (BIST) is a popular solution for the test access/bandwidth problem. The test algorithms generated by TAGS-PS are especially suitable for this purpose, since the simplicity and regularity of test operations can reduce the hardware overhead of the BIST circuit. Moreover, the address generator for March guards can be implemented easily, with a +1 or -1 on the row/column addresses.

Table 3: Port constraints and testing time.

Total ports \ Test ports	6	5	4	3	2
6	25N	25N	25N	33N	61N
5	-	21N	21N	21N	41N
4	-	-	17N	17N	25N
3	-	-	-	13N	13N

7. CONCLUSION

A test algorithm generation and port scheduling methodology for multi-port memories is proposed in this paper. The notion of simulation-based test algorithm generation, March guards, and port scheduling is introduced and discussed in detail. The resulting tests are efficient and cost-effective. The regularity and simplicity of TAGS-PS make it especially suitable for generating test algorithms for built-in self-test (BIST) implementations. Port scheduling provides not only effective pattern compaction but also be able to make trade-offs between test bandwidth and testing time. Our future work includes supporting more fault models, diagnosis algorithm generation, and BIST designs for multi-port memories.

8. REFERENCES

[1] M. Nicolaidis, V. Castro Alves, and H. Bederr, “”testing complex couplings in multiport memories””, *IEEE Trans. VLSI Systems*, vol. 3, no. 1, pp. 59–71, Mar. 1995.

[2] A. J. van de Goor S. Hamdioui, “Fault models and tests for two-port memories”, in *Proc. IEEE VLSI Test Symp. (VTS)*, 1998, pp. 401–410.

[3] T. Matsumura, “An efficient test method for embedded multi-port RAM with BIST circuitry”, in *Proc. IEEE Int. Workshop on Memory Technology, Design and Testing (MTDT)*, 1995, pp. 62–67.

[4] Y. Wu and S. Gupta, “Built-in self-test for multi-port RAMs”, in *Proc. Sixth IEEE Asian Test Symp. (ATS)*, 1997, pp. 398–403.

[5] J. Zhao, S. Irrinki, M. Puri, and F. Lombardi, “Detection of inter-port faults in multi-port static RAMs”, in *Proc. IEEE VLSI Test Symp. (VTS)*, 2000, pp. 297–302.

[6] C.-F. Wu, C.-T. Huang, K.-L. Cheng, and C.-W. Wu, “Simulation-based test algorithm generation for random access memories”, in *Proc. IEEE VLSI Test Symp. (VTS)*, Montreal, Apr. 2000, pp. 291–296.

[7] C.-F. Wu, C.-T. Huang, C.-W. Wang, K.-L. Cheng, and C.-W. Wu, “Error catch and analysis for semiconductor memories using March tests”, in *Proc. IEEE Int. Conf. Computer-Aided Design (ICCAD)*, San Jose, Nov. 2000, pp. 468–471.

[8] C.-T. Huang, J.-R. Huang, C.-F. Wu, C.-W. Wu, and T.-Y. Chang, “A programmable BIST core for embedded DRAM”, *IEEE Design & Test of Computers*, vol. 16, no. 1, pp. 59–70, Jan.-Mar. 1999.

[9] C.-F. Wu, C.-T. Huang, and C.-W. Wu, “RAMSES: a fast memory fault simulator”, in *Proc. IEEE Int. Symp. Defect and Fault Tolerance in VLSI Systems (DFT)*, Albuquerque, Nov. 1999, pp. 165–173.

[10] A. J. van de Goor, *Testing Semiconductor Memories: Theory and Practice*, John Wiley & Sons, Chichester, England, 1991.