A Serial Interfacing Technique for Built-In and External Testing of Embedded Memories

Benoit Nadeau-Dostie*, Allan Silburt*, Vinod K. Agarwal**

*Bell-Northern Research Ltd., 185 Corkstown Road, Nepean, Ontario, Canada, K2H8V4
** McGill University, 3480 University Street, Montréal, Canada, H3A 2A7

ABSTRACT

This paper describes the design and implementation of a test access method and built-in self-test scheme for embedded static RAMs based on a new serial interfacing technique.

INTRODUCTION

Various BIST algorithms and techniques have been proposed [1-6,9,11] to test RAMs. All of the techniques cited evaluate all the bits of each memory word in parallel as it is read. A comparator or compressor module is then usually used for each bit of the output data path. This may be costly in terms of area for wide data paths. Some of the BIST schemes [1,4-6,9] require modifications of the RAM circuitry which may not be possible or practical in many ASIC design environments. When signature compression of the data is used [4,6,9], it adds the problem of aliasing uncertainties to the calculation of fault coverage [8]. There is also a significant problem in applying all schemes cited above when multiple embedded RAMs of varying sizes and port configurations are used on one chip. If each memory requires its own dedicated BIST circuit, the chip area devoted to testing can be unacceptably high. Therefore, it is advantageous to share the BIST circuit among several RAM blocks. It is also desirable to re-use as much of the test circuitry as possible in the normal operation of the chip. It can be quite difficult or inefficient to achieve this sharing of circuitry with the existing schemes.

This paper describes a serial interfacing technique for embedded RAMs which has been successfully applied to static single port and dual port memories in custom integrated circuits. A single bit of the input data path of a RAM (or a group of RAMs) is controlled by the BIST circuit and a single bit of the output data path is observed during the execution of the algorithms. The other bits are controlled and observed indirectly through the serial data path.

Automatically generated BIST circuits have been developed around this serial technique which embed an algorithm suited for the application with the RAM. The serial data path interface has also been used to provide external access to memories on cost sensitive chips which could not justify the full BIST overhead. This provides a simple external test access mode which uses a minimal number of pins yet exercises the memory at full speed.

SERIAL ACCESS TECHNIQUE

Figure 1 shows a block diagram of a static RAM memory (inside box with dotted outline) and the additional external connections required to implement the shifting capability. The address bus is not latched and applied directly to the X (row) and Y (column) decoders. The input and output data paths are separate. When the READ strobe is high, a word is read and transfered to the output of the memory. When the strobe goes back low, the data is maintained by the transparent latches located at the output of the sense amplifier until the next read operation. When the WRITE strobe is high, a word is written into memory at the location determined by the address.

The shifting capability is provided by the multiplexers connected along the I/O data path. The multiplexers select either the normal inputs or the test inputs depending on the value of the Testmode signal. The test input applied to input i of the memory is simply output i-1 or a signal controled directly by the BIST circuit in the case of the least significant bit. Suppose that M_i and M_{i+1} are two logically consecutive bits of the same word. They may or may not be physically adjacent in memory depending on the layout. In order to move the contents of M_i to M_{i+1}, a read operation of the word containing these two bits is first performed bringing the bit in M_i to the corresponding output latch. A write operation is then performed at the same location to store the bit in M_{i+1}. Actually, the full word gets shifted by one position. The memory cell storing

IEEE 1989 CUSTOM INTEGRATED CIRCUITS CONFERENCE CH2671-6/89/0000-0136 $1.00 © 1989 IEEE

the least significant bit of the word has been written with the data provided by the BIST circuit (serial input). The bit originally in the cell storing the most significant bit is now available at the corresponding output latch only and must be examined before we can perform a read operation at any location including the one we just read. This output is called the serial output of the memory and is the only output available to the BIST circuit. Successive read, write, read, write, ... operations on a given address serially shifts out the contents of the word, replacing it with new data from the serial input. The possibility of changing a single bit at a time allows one to increase the coverage of coupling faults between bits of the same word that are physically adjacent in the memory. Also, a single comparator can be used to analyse the response of the memory under test.

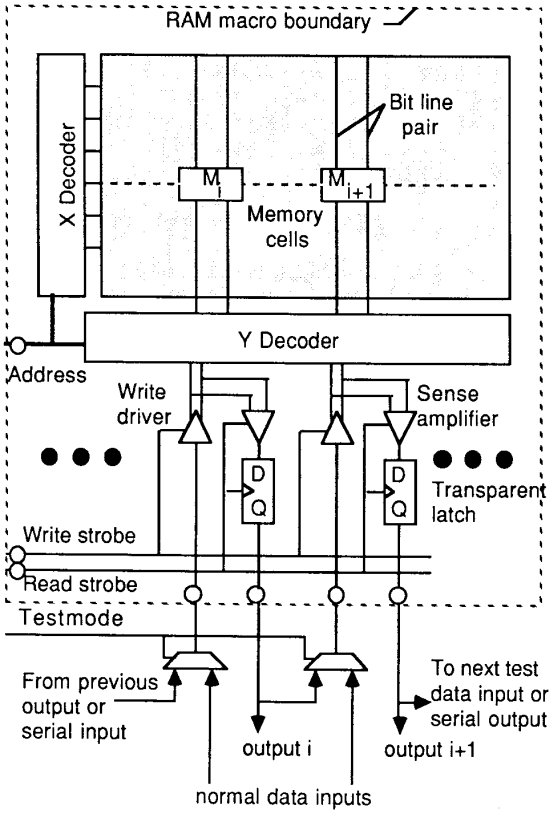

Figure 1 Simplified memory showing serial data path connections for BIST mode.

Other serial access mechanisms have been reported [1,5]. The one described here differs from [5] in that it does not require any modification of the RAM itself or special assumptions about its operation. It requires less circuitry than in [1] where full flip-flops are placed on the data path to provide the same capability. In our approach, the shift register is built with the memory cells themselves and the data output latches.

Since a read and write cycle is performed for each shift operation, this serial technique does not compromise the speed at which new data is applied to the memory. With the algorithms described below, high coverage external tests can be applied at full speed. Another benefit is the possibility the serial mode to test logic external to the memory by linking it to a regular scan path or building an LFSR (Linear feedback Shift Register), or MISR (Multiple Input Signature Register) with the shift register.

BIST ARCHITECTURE AND IMPLEMENTATION

The general architecture for a BIST circuit employing the serial technique is shown in Figure 2. It consists of the following components:

1) Data path multiplexers to set up the shifting mode.

2) Multiplexers on the address and control lines to select normal or test mode access to the RAM.

3) A set of counters whose length depends on the RAM dimensions and the algorithm being used. In all cases a counter is required which has the same number of states as memory addresses (address counter) and another with the same number of states as the bits in a single word (position counter).

4) A finite state machine (FSM) which embodies the actual test algorithm. This circuit controls the counters, generates the serial data stream and the expected data for comparison with the RAM output.

5) A memory control timing generator

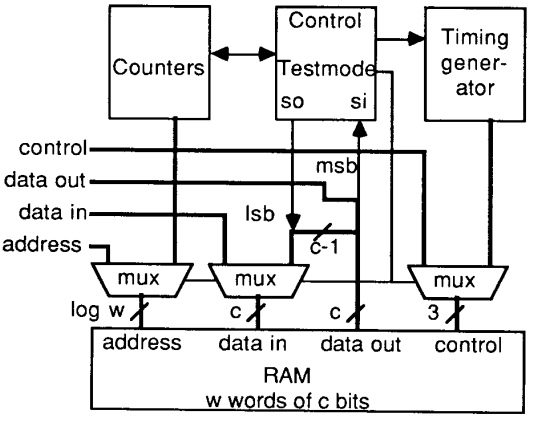

Figure 2 Overall architecture of a BIST circuit

The hierarchy of the circuit is so arranged that blocks such as counters which may also be useful in mission mode are separate. The finite state machine which determines the algorithm to be used is also in a block of its own. This architecture also facilitates the sharing of the BIST circuit between several small embedded RAM blocks. The key issues in how easily a BIST circuit can be shared are:

1) How much interconnect is required between the BIST circuit and the individual RAM blocks

2) How similar the memories are

3) How much additional circuitry is required

4) How easily an ASIC designer can assemble the network from simple guidelines.

The serial data path inherantly reduces the interconnect complexity particularly when the RAMs have large word widths. For all of the proposed algorithms, the address and control signals are applied simultaneously to all memories to be tested. Therefore these lines are common for all RAM blocks again simplifying the interconnect. Two schemes for sharing the BIST circuit may be possible depending on the algorithm: daisy-chaining and test multiplexing. The first consists of simply connecting the serial output of one memory to the serial input of another so they appear as one larger block as shown in Figure 3. As mentioned above, the address and control signals are applied simultaneously to all RAMs. For most algorithms, this method requires that all of the memories have the same number of words, although the number of bits per word is not restricted. Only the position counter needs to be set to the word width sum in this case.

When test multiplexing is used, a single memory is tested at a time. The address and position counters are reconfigured for each memory and the appropriate serial output coming back of the memory is selected making this method a little more complex but more versatile than the daisy-chaining method.

ALGORITHMS

Four algorithms were considered for implementation using this serial access method: SMARCH, SMARCHDEC, SGALPAT and SWALK. As the names suggest, they are adaptation of well-known algorithms. The adaptation mainly pertains to the serial nature of the interface to the memory but also involves an extension of the conventional fault models to include, for example, bit coupling within a word. Stuck-open faults in static address decoders, open access transistors and stuck word lines also require special considerations [10].

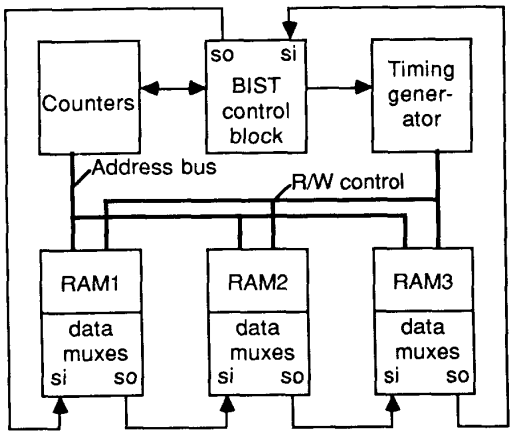

Figure 3 Daisy chain connection for sharing a BIST circuit among 3 memories.

As an example, the SMARCH algorithm is described in Figure 4. This is a March-like test similar to algorithm "C" described in [7]. The notation used to describe the algorithm is as follows. The address space has "w" words and each word contains "c" bits. A read operation is noted as R0, R1 or Rx depending on the expected value at the serial output (x = "don't care"). For a write operation, the terms W0 or W1 are used. Only the serial input is forced to the value indicated. Repetitive operations on a particular word are described by an expression such as $(R0W1)^c$. This set of operations shifts out c 0s from a word and replaces them with c 1s. For each step of the algorithm, a second set of operations such as $(R0W0)^c$ or $(R1W1)^c$ is simply "refreshing" the value already stored and is necessary because of the extensions to the fault model. The total number of read plus write operations is 24 cw, that is, 24 times the number of bits in the memory.

```
1) FOR address 1 to w      {count address forward}
     (RxW0)^c (R0W0)^c     {initialize memory with 0s}
2) FOR address 1 to w
     (R0W1)^c (R1W1)^c     {read 0s and replace with 1s}
3) FOR address 1 to w
     (R1W0)^c (R0W0)^c     {read 1s and replace with 0s}
4) FOR address w to 1      {reverse address sequence}
     (R0W1)^c (R1W1)^c     {read 0s and replace with 1s}
5) FOR address w to 1
     (R1W0)^c (R0W0)^c     {read 1s and replace with 0s}
6) FOR address w to 1
     (R0W0)^c (R0W0)^c     {final RAM state is selectable}
                           {by changing the polarity of }
                           {last write operation by W1}
```

Figure 4 Pseudo-code for algorithm SMARCH

The SMARCHDEC algorithm is simply an extension of the SMARCH test described above. It tests stuck-open faults which were observed in the static address decoder used in some CMOS memories. Certain applications do not exercise all address transitions so they are less sensitive to this failure mode. Therefore we use this as an extension to the SMARCH algorithm for susceptable memories only on an application specific basis since it adds about 30% to the BIST area overhead.

Table 1 lists the various attributes of the four algorithms. The coverage of unmodeled faults is estimated based both on test chips and partial analysis of multiple faults. The bracket for the nominal area overhead (i.e. without sharing) is about 30%. For example it takes about 30% more area to implement SMARCHDEC than SMARCH for a typical memory (4 Kbits). The BIST circuitry is easy to share among several memories for all tests. This was one of the main objectives of this work. Finally, some of the algorithms require counters which are easier than others to re-use in the normal operation of the chip. Generally speaking the SMARCH algorithm is the most widely applicable. However, when the test time is not prohibitive, the SGALPAT is used since it is a more exhaustive test.

Comparison criteria	Smarchdec		Sgalpat	
	Smarch			Swalk
test/simulation time	O(cw)	O(cw)	$O(cw^2)$	$O(cw^2)$
decoder stuck-open fault coverage	No	Yes	Yes	Partial
unmodeled faults coverage	Low	Low	High	Med
area overhead (no sharing)	Low	High	Med	Low
ease of sharing BIST among RAMs	High	High	High	High
ease of re-use of BIST counters	Med	Low	Med	High

Table 1 Comparison of serial test algorithms

IMPLEMENTATION RESULTS

The area overhead for the BIST circuit implementing the SMARCH algorithm is illustrated in figure 5. The areas are quoted for an auto-routed standard cell implementation of the BIST which includes all of the circuitry listed above. The upper curve is for a dedicated BIST circuit for a single memory. Significant BIST area is saved if the address and position counters are re-usable in mission mode. This is shown by the second curve (black diamonds). The last curve (black squares) shows the improvement in efficiency if two identical RAMs share the same BIST circuit. This curve is valid if the memories are nearby or if they share a common address bus. Otherwise, an extra routing factor must be taken into consideration. For all curves, it is clear that the BIST area decreases rapidly with the size of the memory.

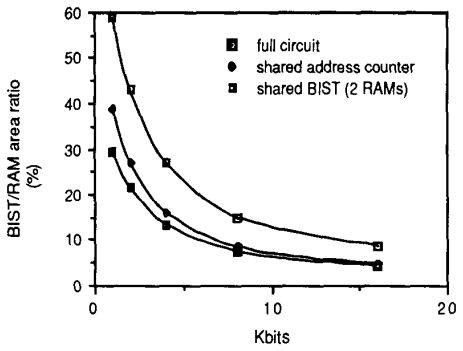

Figure 5 SMARCH BIST circuit area overhead

The effectiveness of the test algorithms and BIST methodology were verified using a test chip. Subsequently the scheme has been successfully applied to several product chips implemented in a 1.5 µm CMOS process. Table 2 shows three specific examples. The same algorithm (SMARCHDEC) is used in all the cases. The BIST counters were not needed in mission mode so the entire BIST area is considered in the fourth column. In all cases, daisy-chaining was used to share the BIST circuitry.

Chip #	RAM configuration	Total bits	BIST / Chip area	Test time
1	1 : 64 x 32 (S)	2,048	1.8%	25 ms
2	1 : 512 x 10 (D) 1 : 512 x 5 (S)	7,680	3.4% †	200 ms
3	2 : 512 x 8 (S) 2 : 512 x 16 (S) 2 : 64 x 8 (D)	25,600	4.3% †	400 ms

S = single-port static RAM
D = dual-port static RAM
† 2 BIST circuits used
configuration = # of memories : # of words X word width

Table 2 Implementation examples

AKNOWLEDGEMENT

The authors want to thank M. Hojjat Salemi for his valuable help during the layout of the first test chips and M. Larry Sasaki for pertinent implementation hints.

CONCLUSIONS

A serial interfacing technique for testing embedded RAMs has been presented. It is applicable to external as well as built-in self-test. The required interconnect compared to parallel test schemes reduces the complexity and silicon area dedicated to testing as well as the number of pins required for external access. This is particularly powerful in a telecom ASIC environment where it is common to find a variety of small RAMs of different sizes and port configurations, some with very wide word widths. Since BIST circuitry is easily shared among several embedded RAMs and in many cases re-usable in the normal of operation of the chip, this method enables on-chip testing with minimal area overhead compared to other schemes. Several algorithms have been adapted to the serial format. The selection of the algorithm within a standard BIST architecture allows the designer to make trade-offs between fault coverage requirements, test time, re-use of the BIST circuitry and area overhead. The generation of the BIST circuitry has been automated and deployed on several product chips. The area overheads were found to be acceptable and have promoted the use of BIST as a means of assuring testability of chips with embedded RAMs.

REFERENCES

[1] Sun Z., Wang L.T., "Self-testing of embedded RAMs", Proc. IEEE International Test Conference, pp. 148-156, October 1984.

[2] Nicolaidis M., "An efficient built-in self-test scheme for functional test of embedded RAMs" , Proc. IEEE International Test Conference, pp. 118-123, November 1985.

[3] Bardell P.H., McAnney W.H., "Self-test of Random Access Memories", Proc. IEEE International Test Conference, pp.352-355, November 1985.

[4] Sridhar T., "A new parallel test approach for large memories", IEEE Design & Test, pp.15-22, August 1986.

[5] You Y., Hayes J.P., "A self-testing dynamic RAM chip", Proc. MIT Conference on Advanced Research in VLSI, pp. 159-168, January 1984.

[6] Jain S.K., Stroud C.E., "Built-in self testing of embedded memories", IEEE Design & Test, pp. 27-37, October 1986.

[7] Marinescu M., "Simple and efficient algorithms for functional RAM testing", Proc. IEEE International Test Conference, pp. 236-239, November 1982.

[8] Smith J.E., "Measure of the effectiveness of fault signature analysis", IEEE Transactions on Computers, vol. C-29, no. 6, pp. 510-514, June 1980.

[9] Han S.H., Malek M., "Two-dimensional multiple-access testing technique for random-access memories", Proc. of IEEE International Test Conference, pp. 248-251, November 1986 .

[10] Dekker R., "Fault modeling and Self-Test of Static Random Access memories", Master Thesis, DELFT University of Technology, 80 pages, September 1987.

[11] Saluja K. K., Eng S. H., Kinoshita K. , "Built-in self-testing RAM: A practical alternative", IEEE Design & Test, pp. 42-51, February 1987.