VLSI Design
1994, Vol. 1, No. 4, pp. 327–334
Reprints available directly from the publisher
Photocopying permitted by license only

STD Architecture: A Practical Approach to Test M-Bits Random Access Memories

ROCHIT RAJSUMAN
Department of Computer Engineering and Science, Case Western Reserve University, Cleveland, OH

KAMAL RAJKANAN
Zilog Inc., Campbell, CA

We present a design method (called STD architecture) to design large memories so that the test time does not increase with the increasing size of memory. Large memories can be constructed by using several small blocks of memory. The memory address decoder is divided into two or more levels and designed such that during the test mode all small memory blocks are accessed together. With the help of modified decoder, all small memory blocks are tested in parallel using any standard test algorithm. In this design, time to test the whole memory is equal to the time required to test one small block. The proposed design is highly structured and hardware overhead is negligible. The basic idea is to exploit internal hardware for testing purpose. With the proposed method a constant test time can be achieved irrespective of the memory size. STD architecture is applicable to memory chips as well as memory boards, and the design is suitable for fault detection as well as for fault diagnosis.

Key Words: *Design for Testability; Memory Testing; RAM Testing; Fault Detection; Fault Diagnosis*

Memory is a very important part of a computer system. Significant amount of work has been done in the recent years to obtain fast and very large memory systems. As a result, the density of semiconductor memory chips has increased dramatically [1]. With the increasing complexity, it has been recognized that the efficient testing of such memories is a difficult problem. A multi-mega bit random access memory requires excessively large time just to test all cell stuck-at faults. To overcome this problem, researchers have sought to develop innovative test generation algorithms and on-chip built-in self-test methods.

Several innovative test algorithms for random access memories have been reported [2–17]. These algorithms can be categorized into two classes. One set of algorithms is based on the memory fault model as given by Nair, Thatte and Abraham [3]. The representative papers are [3–11]. A second class of test algorithms is based on the pattern sensitive neighborhood cell fault model of Hayes [12–13]. The representative papers are [12–17]. The best known algorithms in both the classes are polynominal in time. Because the complexity of memories is quadrupling in every 2–3 years, even a linear increase in test time becomes undesirable for large memories [1].

To overcome the problem of large test time, built-in self-test (BIST) methods have been developed. One set of papers use extra hardware for on-chip test generation and response evaluation (using a parallel signature analyzer) [18–22]. The second approach [23–30], uses extra hardware to partition the whole memory into small blocks and test them in parallel (using external test generation). Jarwala and Pradhan [25], showed that using partitioning methods, a significant saving in the test time can be achieved for large memories. However, Jarwala and Pradhan [25] have pointed out that if a 1M-bit memory is partitioned into 16 blocks (64K-bit modules) using H-tree, the percentage increase in area is about 30%. Such a large overhead in hardware is unaffordable. It decreases the manufacturing yield significantly as well as causes performance degradation.

Because of large test time requirements, memory testing problems cannot be solved alone by clever test algorithms. We present a structured testable de-

sign method (STD architecture), to build (design) large memories using small blocks. The design allows testing of small blocks in parallel in a manner that testing time for the whole chip remains constant irrespective of the memory size. Because small memory blocks are used to build big modules, the concept is directly applicable to memory board design. In the following section, we have commented on memory fault model and test generation algorithm. The section after that presents the STD architecture.

FAULT MODEL AND TEST GENERATION

A general method to test a system is to assume a fault model and to generate input vectors to cover these faults. A widely used fault model for RAM devices is the fault model of Nair, Thatte and Abraham [3]. In this model, a circuit is divided into three blocks, i.e., memory cell array, decoder circuit and the sense amplifier or read/write circuit. In memory array, a cell may have a stuck-at-1/0 fault or a cell may have a coupling fault with other cells. In the decoder circuit, a decoder may not access the addressed cell, it may access a non-addressed cell or it may access multiple cells. The read/write circuit may have stuck-at-1/0 faults which appear as memory cell stuck-at faults. Physical fault mechanisms in memory devices have been investigated [5], and it was found that all faults in memory could be covered by the fault model given by Nair et al. [3] with the addition of state transition faults and data retention faults. A more general fault model would thus include the following:

1. Memory cell stuck-at-1/0 fault.
2. Memory cell state transition 1-to-0 and 0-to-1 fault.
3. Memory cell bridging with other cell (state coupling).
4. Stuck-at, multiple access or wrong addressing faults in decoder.
5. Data retention fault.

For simplicity, for the test approach presented in section 3, we have assumed the fault model as given in [5]. It should be noted that any fault model can be used with an appropriate test generation algorithm.

Ref. [5] also presents a RAM test algorithm of linear time complexity to cover 100% faults under single fault assumption. The number of read/write cycles is kept to a minimum. The algorithm requires only 9n read/write operations where n is the number of bits. This algorithm covers all cell stuck-at-1/0 faults, 1-to-0 and 0-to-1 state transition faults, state coupling faults between two cells, data retention faults and decoder faults. In this paper, we assume the fault model and test algorithm as given in [5]. Due to space limitation, test algorithm has not been reproduced here, readers are referred to [5]. It should be noted that STD Architecture as given in section 3, is not limited to any specific fault model or test algorithm. Depending upon objectivity, any fault model and test algorithm can be used with STD Architecture.

STD ARCHITECTURE

The basic philosophy behind the proposed architecture is to divide an entire memory into several small blocks and test the blocks in parallel. This partitioning is done during the design phase itself. The main idea is to partition the memory address decoder into multi-levels and design the memory accordingly [27–28]. This partitioning method does not need large hardware overhead and only requires a modified decoder for the most significant address lines to ensure testability. As will be demonstrated, the proposed decoder modification does not cause large hardware overhead. The basic difference between the proposed scheme and that of Jarwala and Pradhan [25], is that the latter partitions memory using extra hardware (making H-tree) after the design. The access to the partitioned blocks was obtained by extra hardware. In the proposed STD architecture, partitioning is achieved through memory address decoder.

Below, we have used four examples to illustrate the proposed scheme. These examples cover both 1-bit and m-bit word size memories, and allow memory size to increase by a factor of 2^x. To express the memory size, we have used the notation nxm, where n represents the number of words and m represents the word size. For example, 16K × 1 represents 16K-bits and 16K × 8 represents 16K-bytes.

Example 1 8K × 1 Memory

This memory requires a 13-to-8K decoder which can be implemented at two levels, using 10-to-1K decoder and a 3-to-8 decoder. Thus, the memory can be designed by eight blocks of 1K-bits (each associated with 10-to-1K decoder) and a 3-to-8 decoder.

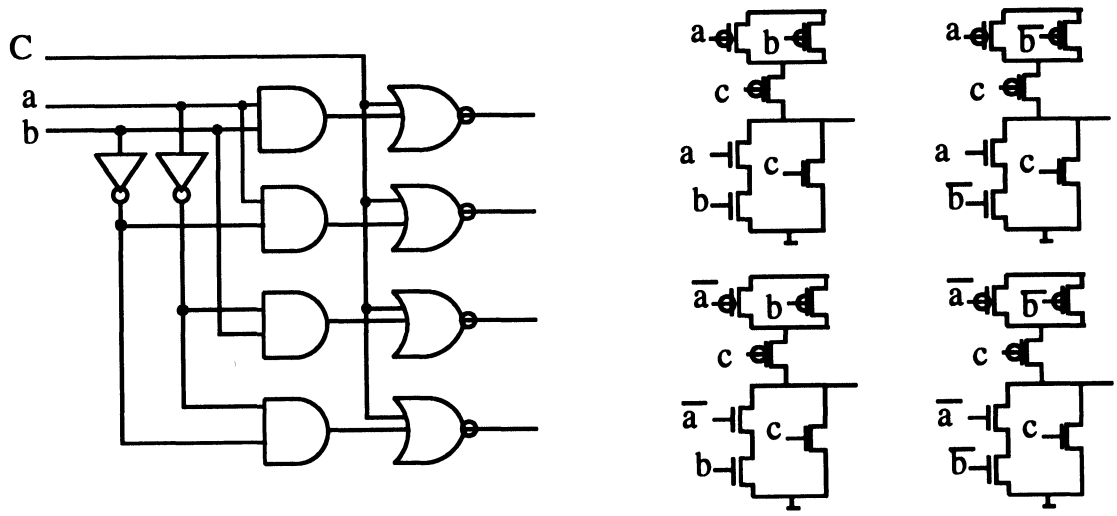

FIGURE 1 Circuit diagram of modified 2-to-4 decoder with an external control c.

The design is made testable by modifying the 3-to-8 decoder, which contains the most significant address lines A10-A12. The modification is done by adding one extra control signal to the decoder. To illustrate the concept, design of 2-to-4 decoder is given in Figure 1. Also, a parity circuit is added at the outputs of the 1K-bits blocks. The testable design is given in Figure 2. The control signal added to the 3-to-8 decoder makes it possible to select all of the decoder output lines when control signal C = 1 (this is done during the test mode). When C = 0, the decoder is in its normal mode and selects only one of its output.

To test this memory, the control signal C is kept at '1.' Thus, same data read/write operations can be done to all eight memory blocks using address lines A0–A9. During this mode, all eight block are tested in parallel. It should be noted that in case of a fault in any block, the output of the parity circuit would be '1' and hence, fault is detected. Using the algorithm as given in [5], all eight blocks can be tested by 9K read/write operations.

After the testing of memory blocks, the control C is switched to '0', converting the 3-to-8 decoder into normal mode. Under this situation, eight input combinations are needed to test the 3-to-8 decoder. It should be noted that if 1K×1 blocks are tested by the algorithm given in [5], all cells contain '0' after the conclusion of test. Hence, for each combination of address lines A10–A12, we can detect decoder faults by writing a '1' in a cell and reading it. However, to test the next combination, the cell should contain '0'. Thus, 24 read/write operations are required (last write is not necessary, thus, 23 operations are enough). These eight possible combinations are applied at the address lines A10–A12, while keeping A0–A9 to a fixed value (preferable, all 0s or all 1s). The response is observed at D_{out} line. Thus, the whole 8K-bits memory can be tested in a test time necessary to test 1K-bits memory, (9K + 24) read/write operations to be accurate. The hardware overhead in this design is one control signal, 4 XOR

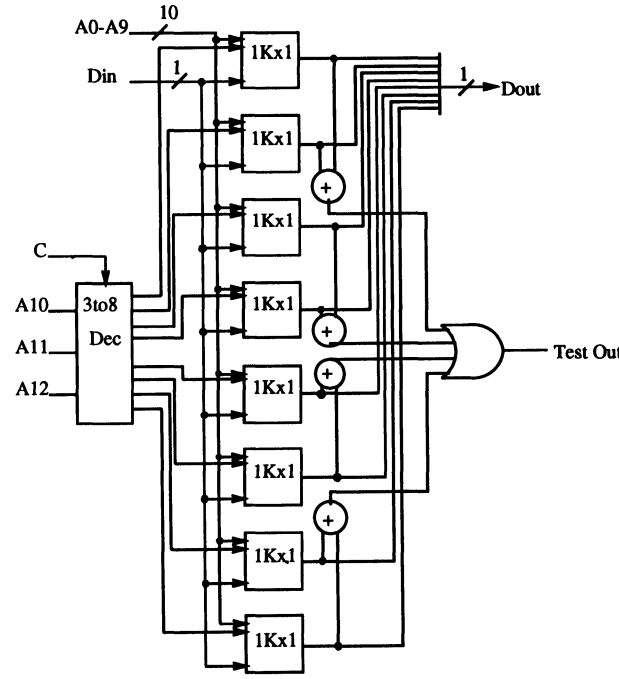

FIGURE 2 Testable design of 8K×1 memory for example 1.

gates and 1 OR gate. Because the control line is limited to the 3-to-8 decoder and does not extend to the memory blocks, the routing area is neglible for the control signal. The 10-to-1K decoders within the memory blocks can also be implemented in two- or multi-level. However, these decoders need no modification because partition size is 1K-bits.

It should also be noted that this design does not cause any performance penalty. Breaking down a single stage large decoder into multi-level decoder is in fact desirable, because total delay of multi-level decoder is small due to smaller capacitances.

Extension in Memory Size

The memory size can be extended in this architecture with minimal effort. For example, consider the design given in example 1 is required to be extended to 32K-bits memory. This can be done easily by using four blocks of 8K-bits memories each equivalent to Example 1, and an additional 2-to-4 decoder. This 2-to-4 decoder contains the most significant address lines and hence, it is modified by a control signal. The control signal used in the 3-to-8 decoder in Fig. 3 can be used in 2-to-4 decoder as shown in Figure 3. During the test mode, the four 8K×1 blocks are selected using the control signal (C = 1) and all eight 1K×1 blocks are selected for each 8K×1 block. Therefore, by setting C = 1, all 32 blocks of 1K×1 memory are selected. These 32 blocks are tested in parallel by 9K read/write operations using address lines A0–A9.

After testing of the memory blocks, the control signal is switched to '0'. Under this condition, the 2-to-4 decoder and four 3-to-8 decoders are tested by 32 combinations (96 read/write operations). Hence, whole 32K-bits memory is tested as 1K-bits memory using (9K + 96) vectors. If the control signals to 2-to-4 decoder and 3-to-8 decoders are different, then all four 3-to-8 decoders can also be tested in parallel. In that case, total number of test vectors is (9K + 36).

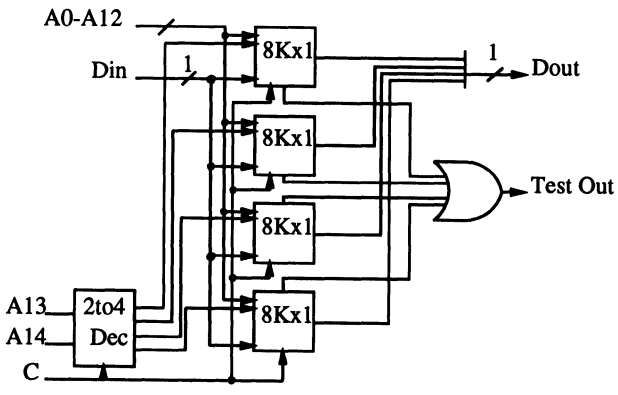

FIGURE 3 Testable design of 32K×1 memory. Each 8K×1 block is equivalent to Figure 2, producing 1-bit outout. Output lines of 2-to-4 decoder behave as chip select signal for 3-to-8 decoder inside 8K×1 blocks.

Extra hardware required in this 32K×1 design is four parity circuits, each having 4 XOR gates and 1 OR gate, another OR gate and a control signal. The routing area of the control signal is again negligible. Effectively, hardware overhead in this case is 16 XOR gates and 5 OR gates.

The above discussion is restricted to bit-oriented memory where the data line is only one bit wide and observability is poor. When the word size is m-bits, m data lines are observable. Therefore, we can easily divide the whole memory into m blocks and test them in parallel.

Extension to Memory Word Size

The STD architecture allows the extension in memory word size in a straight forward fashion. For example, consider that we need to extend the design of example 1 into byte format. This 8K×8 memory can be designed by using eight blocks of 8K×1 memory (Figure 4). In this design, each block of 8K×1 memory is equivalent to the circuit given in Fig. 2.

All eight blocks of 8K×1 memory can be tested in parallel by making $D_{in0} = D_{in1} = \ldots = D_{in7}$. Effectively, this testing procedure is the same as in Example 1. The number of memory operations is (9K + 24). Because each test output is separately observable, any fault can be detected and faulty block can be uniquely identified. After testing of memory blocks, eight additonal write and read operations (16 operations) are used to test the coupling among D_{in} lines or among D_{out} lines. This is achieved by keeping one D_{in} line at '1' while all the others are at '0', for a fixed value of A0–A12. Thus, the whole 8K×8 memory is tested by (9K + 40) vectors.

The hardware overhead in this case is eight parity trees each having 4 XOR gates and 1 OR gate (total 32 XOR gates and 8 OR gates), and one control signal. It should be noted that modification in the test algorithm to byte-oriented memory using technique such as given in [5] is not desirable. The amount of hardware overhead will remain the same, however, the number of test vectors will increase to (27K + 24).

The following example uses this concept and illustrates the case when an extension in numbers of words as well as in word size is required.

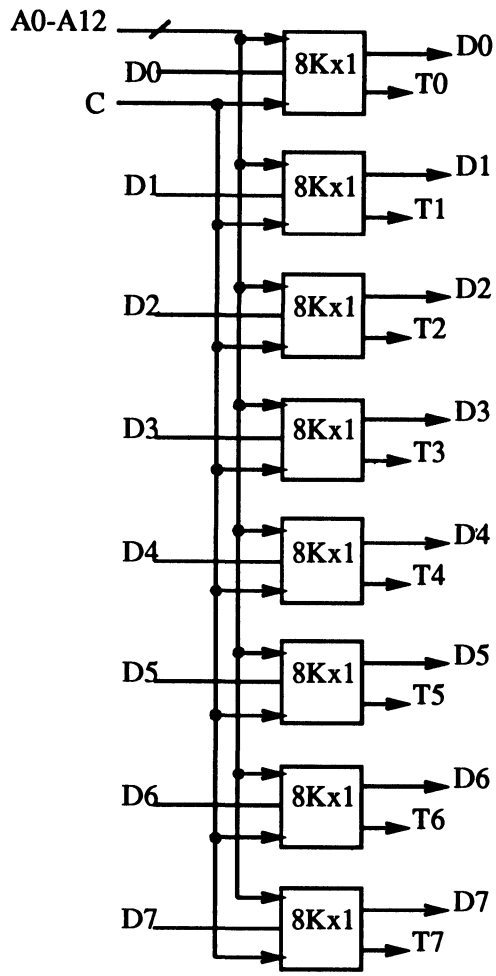

FIGURE 4 Testable design of 8K×8 memory. Each 8K×1 block is equivalent to Figure 2.

Example 2 32K×8 Memory

This memory can be built by using four blocks of 8K×8 memory and a 2-to-4 decoder. The concept is similar to Example 2. In this case, each block of 8K×8 memory is equivalent to Fig. 4. The additional 2-to-4 decoder, which contains the most significant address lines, is modified by adding a control signal. If we consider a separate control signal than used in 8K×8 blocks, the memory can be tested by (9K + 48 + 12) vectors or read/write operations. The hardware overhead in this case is two control signals and [4×(32 XOR + 8 OR) + 8 OR = 128 XOR + 40 OR] gates.

Another possibility is to design this memory by using eight blocks of 32K×1 memory. The concept is similar to the previous 8K-bytes example. However, in this case, each block of 32K×1 memory is equivalent to Fig. 3. With this design, a 32K-bytes memory can be tested by (9K + 96 + 12) vectors. The hardware overhead is one control signal and [8×(16 XOR + 5 OR) = 80 XOR + 40 OR] gates.

In both the approaches, the hardware overhead and test time is comparable. The amount of hardware overhead can be reduced significantly by using bigger blocks of memory arrays instead of 1K×1 arrays. However with the larger blocks, memory test time will increase. For example, if 8K×1 memory blocks are used, the whole memory can be tested by (72K + 12) vectors, while only (16 XOR + 8 OR) gates are required (see Figure 5). It should be noted that equivalently, 256K×1 memory can be designed by four additional 3-to-8 decoders, and can be tested by (72K + 48) read/write operations.

The above examples show that various size and word length memories can be designed such that the test time remains constant in all the cases. From Example 1 for total capacity of 8K-bits, to Example 2 for total capacity of 256K-bits the test time is constant (approximately 9K vectors).

FAULT DIAGNOSIS AND RECONFIGURATION

With slight modifications in the designs given in the preceding section, memory can be designed for fault diagnosis. The basic idea is to use a register instead of a parity circuit to obtain better observability. Consider the design of 8K×1 memory as given in Example 1. The modified design is shown in Figure 6. The design given in Figure 6 is basically same as given in Figure 2, except the OR gate in parity circuit of

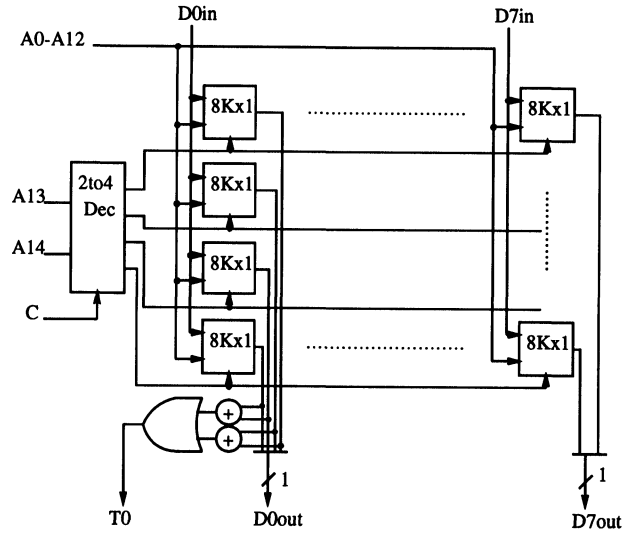

FIGURE 5 Testable design of 32K×8 memory using 8K×1 blocks.

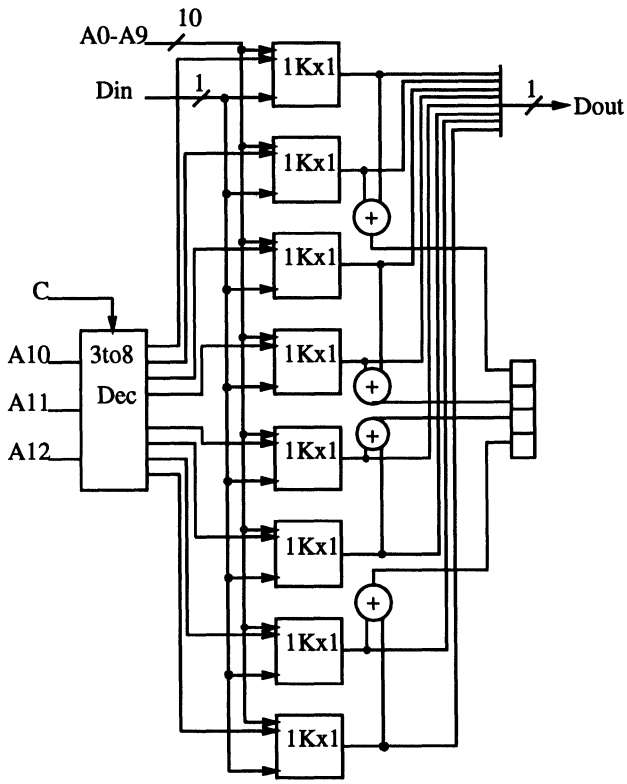

FIGURE 6 8K×1 memory for fault diagnosis and reconfiguration.

Figure 2 has been replaced by a 4-bit register. The test method is same as given in section 3. However, instead of looking one bit parity output, collective response of two blocks (each 1K×1) is observable. Observability can be further enhanced if the whole parity circuit is replaced by an 8-bit register. In that case, a faulty 1K×1 block can be uniquely identified.

In the design of a 32K×1 memory (Figure 3), a four bit register can be used at the output lines of 8K×1 blocks instead of the OR gate. In this case, a faulty 8K×1 block can be identified. If a detailed fault diagnosis is required, four registers can be used within the 8K×1 blocks instead of parity trees. Another possibility is to use a single 32-bits long register in which each bit represents the D_{out} line of 1K×1 memory block. In both the cases, a faulty 1K×1 block can be identified. In general, if only fault detection is required, parity circuits are recommended because the area required by the registers is more compared to the XOR and OR gates.

After locating a faulty block, proposed STD architecture also allows an easy reconfiguration of the memory into 3/4, 1/2 or 1/4 of the original capacity. The idea is to permanently connect the corresponding address line of the most significant part of address decoder to Gnd or Vdd. For example, four blocks (each 1K×1) can be disconnected in Figure 6 by connecting the corresponding address line (A10–A12) to Gnd/Vdd. In comparison to the existing schemes [29–36], reconfiguration in STD architecture is extremely simple and does not require special hardware such as laser programmable switches.

CONCLUDING REMARKS

The main advantage of the proposed scheme is that very small test time can be achieved for any size of memory. The test time can be kept almost constant irrespective of the memory size. The design procedure is highly structured and test vectors need not to be calculated for different memories having same size of partitions (except the additonal vectors needed to test decoders). The design uses existing memory blocks of small size and hence large memories can be designed by this method in significantly small design time. It should also be noted that the architecture is not limited to the chip design. The architecture is directly applicable to memory board design (without any modification).

As discussed with a slight modification, fault diagnosis can also be achieved at module level in this design. Fault diagnosis is extremely important from reliability point of view. In case of memory board, this is highly desirable. By identifying a faulty memory block (memory chip in case of a board), it can be replaced by a good block if redundancy is available. If redundancy is not available, the faulty block can be disconnected and the most significant decoder is modified so that good memory blocks can still be used. The memory can subsequently be configured into 3/4, 1/2 or 1/4 capacity of the original size.

The only negative aspect of the proposed STD architecture is the requirement of extra hardware to observe the test response. For very large memories if the test time is kept extremely small (for example, a 4M-bits memory partitioned into 1K-bits blocks), the hardware overhead becomes significant. The hardware overhead is inversely proportional to the partition size. This trade-off for various memories is shown in Figure 7. It should be noted that in Figure 7, we measure the overhead in the number of gates. Although, this measure is not very accurate, we feel that this is the best representation, because the percentage overhead is negligible. It should be noted that the routing overhead for the control signal is very small in the proposed STD architecture as explained in section 4. The area overhead due to ad-

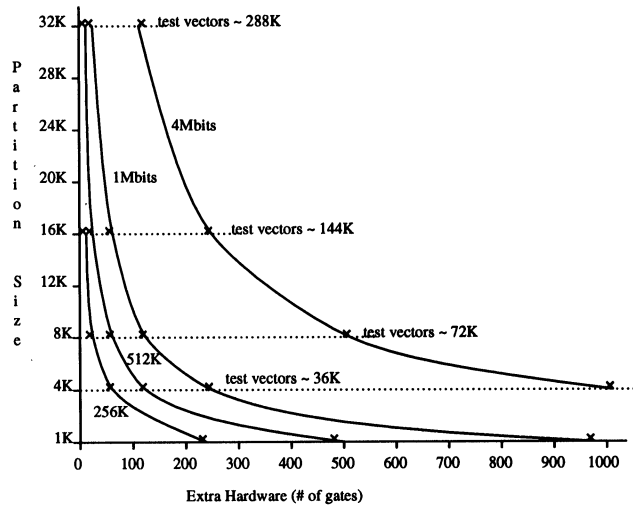

FIGURE 7 Trade-off in partition size vs extra hardware. Dotted lines represent constant test time.

ditional transistors associated with the control signal inside decoder is also extremely small, if the decoder is designed with complex gates as shown in Figure 1. Partitioning and implementation of a decoder into two or multi-level in fact can result in a reduction in transistor count. This fact can be visualized by considering a small example, a 4-to-16 decoder. One level implementation of 4-to-16 decoder using 4-input gates requires 128 transistors (64 nMOS and 64 pMOS transistors). The same decoder can be implemented at two levels using five 2-to-4 decoders. In this case, the total number of transistors is ($5 \times 16 = 80$). Partitioning of the decoder in this manner also results in the decrease in signal propagation delay [37], due to small capacitances. Therefore, such partitioning is desirable to improve the performance.

References

[1] K. Itoh, "Trends in megabit DRAM circuit design," IEEE J. Solid State Circuits, Vol. 25(3), 778–789, June 1990.

[2] J. Knaizuk and C.R.P. Hartman, "An optimal algorithm for testing stuck-at faults in random access memories," IEEE Trans. Comp., Vol. 26(11), pp. 1141–1144, Nov. 1977.

[3] R. Nair, S.M. Thatte and J.A. Abraham, "Efficient algorithms for testing semiconductor random access memories," IEEE Trans. Comp., Vol. 27(6), pp. 572–576, June 1978.

[4] R. Nair, "Comments on an optimal algorithm for testing stuck-at faults in random access memories," IEEE Trans. Comp., Vol. 28(3), pp. 258–261, March 1979.

[5] R. Dekker, F. Beenker and L. Thijssen, "A realistic fault model and test algorithm for static random access memories," IEEE Trans. CAD, Vol. 9(6), pp. 567–572, June 1990.

[6] R. Dekker, F. Beenker and L. Thijssen, "Fault modeling and test algorithm development for static random access memories," Proc. Int. Test Conf., pp. 343–352, 1988.

[7] A. Birolini, W. Buchel and D. Heavner, "Test and screening strategies for large memories," Proc. European Test Conf., pp. 276–283, 1989.

[8] T. Fuja, C. Heegard and R. Goodman, "Linear sum codes for random access memories," IEEE Trans. Comp., Vol. 37(9), pp. 1030–1042, Sep. 1988.

[9] R. Rajsuman, "An algorithm and design to test random access memories," Proc. IEEE Int. Conf. on Circuits and Systems, ISCAS-92, pp. 439–442, 1992.

[10] J. Savir, W.H. McAnney and S.R. Vecchio, "Fault propagation through embedded multiport memories," IEEE Trans. Comp., Vol. 36(5), pp. 592–602, May 1987.

[11] R. David, A. Fuentes and B. Courtois, "Random pattern testing versus deterministic testing of RAMs", IEEE Trans. Comp., Vol. 38(5), pp. 637–650, May 1989.

[12] J.P. Hayes, "Detection of pattern sensitive faults in random access memories," IEEE Trans. Comp., Vol. 24(2), pp. 150–157, Feb. 1975.

[13] J.P. Hayes, "Testing memories for single cell pattern sensitive faults," IEEE Trans. Comp., Vol. 29(3), pp. 249–254, March 1980.

[14] R. Rajsuman. "Algorithms to test pattern sensitive faults in random access memories," IEEE Int. Workshop on Memory Testing, 1993.

[15] S.C. Seth and K. Narayanaswamy, "A graph model for pattern sensitive faults in random access memories", IEEE Trans. Comp., Vol. 30(12), pp. 973–977, Dec. 1981.

[16] P.D. Jong and A.V.D. Goor, "Test pattern generation for API faults in RAM," IEEE Trans. Comp., Vol. 37(11), pp. 1426–1428, Nov. 1988.

[17] P. Mazumder and J.K. Patel, "Parallel testing for pattern sensitive faults in semiconductor random access memories," IEEE Trans. Comp., Vol. 38(3), pp. 394–407, March 1989.

[18] P. Mazumder, J.H. Patel and J.A. Abraham, "A reconfigurable parallel signature analyzer for concurrent error correction in DRAM," IEEE J. Solid State Circuits, Vol. 25(3), pp. 866–870, June 1990.

[19] T. Sridhar, "A new parallel test approach for large memories," IEEE Design and Test, pp. 15–22, Aug. 1986.

[20] S.K. Jain and C.E. Stroud, "Built-in self testing of embedded memories," IEEE Design and Test, pp. 27–37, Oct. 1986.

[21] R. Rajsuman, "Testing of random access memories," chapter 7 in Digital Hardware Testing, Artech House Inc., MA, 1992.

[22] M. Franklin, K.K. Saluja and K. Kinoshita, "A built-in self-test algorithm for row/column pattern sensitive faults in RAMs," IEEE J. Solid State Circuits, Vol. 25(2), pp. 514–523, April 1990.

[23] R. Kraus, O. Kowarik, K. Hoffmann and D. Oberle, "Design for test of Mbit DRAMs," Proc. Int. Test Conf., pp. 316–321, 1989.

[24] P.H. Bardell and W.H. McAnney, "Built-in test for RAMs," IEEE Design and Test, pp. 29–36, Aug. 1988.

[25] N.T. Jarwala and D.K. Pradhan, "TRAM: A design methodology for high performance, easily testable, multimegabit RAMs," IEEE Trans. Comp., Vol. 37(10), pp. 1235–1250, Oct. 1988.

[26] H. McAdams, J.H. Neal, B. Holland, S. Inoue, W.K. Loh and K. Poteet, "A 1-Mbit CMOS dynamic RAM with design for test functions," IEEE J. Solid State Circuits, Vol. 21(5), pp. 635–641, Oct. 1986.

[27] R. Rajsuman, "An apparatus and method for testing random access memories," US patent application, serial number 620359, Nov. 29, 1990.

[28] R. Rajsuman and K. Rajkanan, "An architecture to test random access memories," Proc. 5th Int. Conf. on VLSI Design, Bangalore, India, pp. 144–147, Jan. 1992.

[29] T. Oshsawa, T. Furuyama, Y. Watanabe, H. Tanaka, N. Kushiyama, K. Tsuchida, Y. Nagahama, S. Yamano, T. Tanaka, S. Shinozaki and K. Natori, "A 60-ns 4-Mbit

CMOS DRAM with built-in self test function," IEEE J. Solid State Circuits, Vol. 22(5), Oct. 1987.
[30] P.H. Voss, L.C.M.G. Pfennings, C.G. Phelan, C.M. O'Connell, T.H. Davies, H. Ontrop, S.A. Bell and R.H.W. Salters, "A 14-ns 256K × 1 CMOS SRAM with multiple test modes," IEEE J. Solid State Circuits, Vol. 24(4), pp. 874–880, Aug. 1989.
[31] Y. Matsuda, K. Arimoto, M. Tsukude, T. Oishi and K. Fujishima, "A new array architecture for parallel testing in VLSI memories," Proc. Int. Test Conf., pp. 322–326, 1989.
[32] J.P. Shen, W. Maly and F.J. Ferguson, "Inductive Fault Analysis of MOS integrated circuits," IEEE Design and Test of Computers, pp. 13–26, Dec. 1985.
[33] M.F. Chang, W.K. Fuchs and J.H. Patel, "Diagnosis and repair of memory with coupling faults," IEEE Trans. Comp., Vol. 38(4), pp. 493–500, April 1989.
[34] W.K. Huang, Y. Shen and F. Lombardi, "New approaches for the repairs of memories with redundancy by row/column deletion for yield enhancement," IEEE Trans. CAD, Vol. 9(3), pp. 323–328, March 1990.
[35] C.L. Wey and F. Lombardi, "On the repair of redundant RAMs," IEEE Trans. CAD, Vol. 6(2), pp. 222–231, March 1987.
[36] T. Fuja and C. Heegard, "Row/column replacement for the control of hard defects in semiconductor RAMs," IEEE Trans. Comp., Vol. 35(11), pp. 996–1000, Nov. 1986.
[37] M. Shoji, "CMOS digital circuit technology," Prentice Hall, Englewood Cliffs, NJ, 1988.

Biographies

ROCHIT RAJSUMAN received the B. Tech. degree from K.N. Institute of Technology, India, in 1984, the MS degree from the University of Oklahoma, Norman, OK in 1985 and the Ph.D. degree from the Colorado State University, Fort Collins, CO in 1988, all in Electrical Engineering.

He joined the department of Computer Engineering and Science at Case Western Reserve University, Cleveland, OH, in August 1988 as an Assistant Professor. He also holds a secondary appointment as an Assistant Professor in the Department of Electrical Engineering and Applied Physics, Case Western Reserve University. His research interests include VLSI design, testing, VLSI fault modeling, design for testability, fault tolerant computing, computer architecture and computer networks. He has published more than 35 papers on these topics, has authored "Digital Hardware Testing," Artech House Publishers Inc., June 1992 and co-edited "Bridging Faults and IDDQ Testing," IEEE CS Press technology series volume, Oct. 1992.

Rajsuman is serving on IEEE Technical Committee on Test Technology and Technical Committee on Microprocessors and Microcomputers. He is the chairman of TC Test Technology Memory Technical Activity 1993, and chairman of Speakers Program 1991 and 1992. He has served on the technical program committee of IEEE VLSI Test Symp. for 1991 and 1992. He is the Publicity Chair for the 6th and 7th Int. Conf. on VLSI Design 1993 and 1994, and General Chair for 1993 IEEE Int. Workshop on Memory Testing. Rajsuman is senior member of the IEEE, member of IEEE Computer Soc., IEEE Circuit and System Soc., ACM special interest group Design Automation, Tau Beta Pi and Eta Kappa Nu.

DR. KAMAL RAJKANAN has been active in semiconductor field for last over 18 years. Currently he is with Zilog as the Director of IC Design Methodology. In this capacity, he has been instrumental in establishing a top-down design methodology to apply megacells for quick-turn Application Specific Standard Products. Previously, Dr. Rajkanan managed various technology development and yield enhancement groups at Intel, Performance Semiconductor, Unisys, GTE, and General Instruments leading to various generations of CMOS, BiCMOS, and EPROM technologies. Dr. Rajkanan also co-managed MIMIC Phase I program for Sperry, and cofounded Micromos Inc. He also served on Technology Advisory Board of SRC and MCC. He has been Adjunct Professor at State University of New York, Stony Brook and Polytechnic Institute of New York, Farmingdale. He is the Technical Program Chairman for 1992 IEEE International Workshop on Memory Testing. Dr. Rajkanan holds Ph.D. in Electrical Engineering from State University of New York, Buffalo and has published over 45 papers on various topics of VLSI technology.