Lehigh University
Lehigh Preserve

Theses and Dissertations

1991

A self adapting production system for test time reduction of semiconductor integrted circuit memories

Glen A. Smith
Lehigh University

Follow this and additional works at: https://preserve.lehigh.edu/etd

Part of the Electrical and Computer Engineering Commons

Recommended Citation
Smith, Glen A., "A self adapting production system for test time reduction of semiconductor integrted circuit memories" (1991). *Theses and Dissertations*. 5473.
https://preserve.lehigh.edu/etd/5473

This Thesis is brought to you for free and open access by Lehigh Preserve. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of Lehigh Preserve. For more information, please contact preserve@lehigh.edu.

A SELF ADAPTING PRODUCTION SYSTEM FOR TEST TIME REDUCTION OF SEMICONDUCTOR INTEGRATED CIRCUIT MEMORIES

by

Glen A. Smith

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Computer Science

Lehigh University

1991

A Thesis is presented for partial fulfillment of a masters degree in computer science.

Professor in Charge

Edwin J. Kay

Date 3/6/91

Chairman of Department

Lawrence J. Varenini

Date 3/6/91

Acknowledgements

I thank Professor Edwin Kay for helping in the development of this thesis. He has spent much time reviewing and commenting on the style and content of this paper.

I also thank my wife Sally for her time in reading this document and critical review, and for watching the children so that I had the time to complete this work.

Table of Contents

I. Introduction 2

II. Conventional Techniques for Reducing Test Times 17

1. Reduction of Algorithms 18

2. Test Reordering 19

3. Combinatorial Tests 21

4. Test Removal/Relocation 22

III. Adaptive Test TIme Reduction 24

IV. References 38

V. Appendix A 41

VI. Biography 51

List of Tables

1. Test Times for Algorithms 4

2. Memory Market Share 9

List of Figures

1. Dram Price Per Bit Forcast 3

2. Dram and Sram Cell Structures 7

3. Test Sequence Diagram 15

4. Multi-Tester Production Environment 27

5. Truncated Summary Database 30

6. Signature Analysis 32

7. Typical Test Procedure 35

Abstract

As each generation of integrated circuit random access memory devices quadruples in size, the market price stablizes at approximately the same cost per device. Through the reduction of feature sizes and the increased size of silicon wafers, more devices may be processed per wafer, hence reducing the overall production cost per device. However, with the increased density, the time to test (access), all the elements that make up the memory array is increased. With the cost of state-of-the-art memory testers exceeding $1 million, the throughput of these machines is critical to the final cost of producing a good finished memory device. The amount of time it takes to test one memory device (test time), is crucial to the throughput of the memory tester. Various strategies have been employed in reducing the time it takes to test a good memory device. Some of these strategies are explored with the pros and cons of each explained. Then, an adaptive real-time method for reducing the test time is developed that will maintain the overall acceptable quality level (AQL).

Introduction

Historically, the number of storage cells in an integrated circuit memory has quadrupled roughly every 3.1 (or π) years (A. van de Goor, 1990). The 256Kbit dynamic random access memory (DRAM) was introduced in 1982, the 1Mbit DRAM in 1985, the 4Mbit DRAM in 1988, the 16Mbit DRAM will probably be in production in 1991, and the 64Mbit DRAM can be projected for 1994. As each generation device matures and reaches a high level of production, the cost per device tends to stabilize at approximately the same level. Figure 1 is a DRAM price/bit forecast (McClean, 1990). These costs are achievable when the device is in full production, the manufacturing process is well understood and under control, and the yields are high. One thing that is hidden in this table is that as the density quadruples (each generation of device has four times the number of storage cells as the previous generation), and the complexity increases, the test cost per device, which is a function of the test time, may not significantly increase.

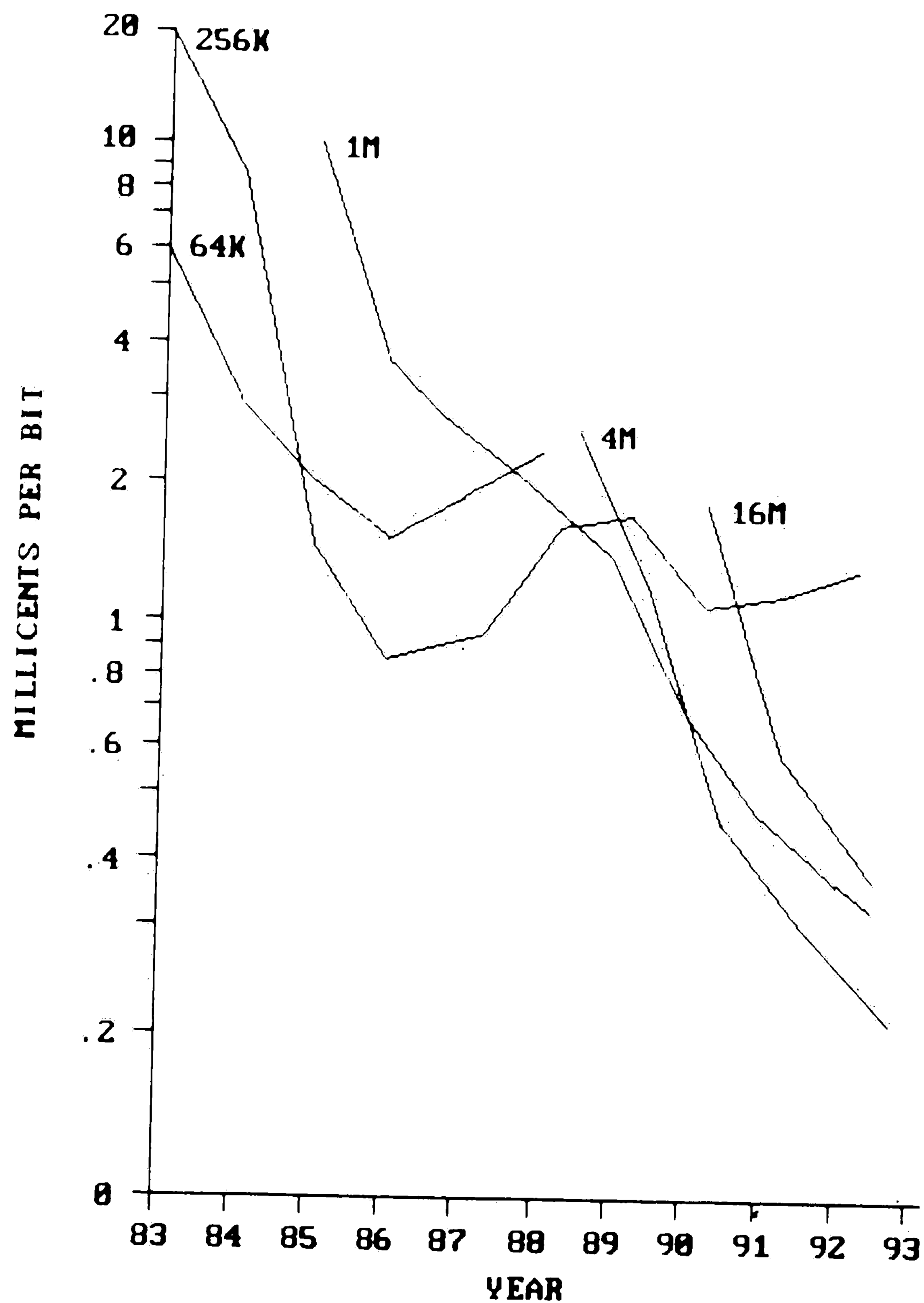


Figure 1. DRAM Price Per Bit Forecast

What defines a thorough test will be covered later in this paper, but for now we will define a test as a series of algorithms that judge the ability of the device to store and retrieve data. In the early days of low density devices these algorithms often took test times in the order of O(n.log(n)) or even O(nxn), where n is the number of storage cells in the device. The

large O notation is an important concept in the expression of theoretical efficiency of an algorithm and is known as asymptotic notation (Brassard & Bratley, 1988). An expression such as O(nxn), is stated as "in the order of n squared". When used in the context of memory test times, it signifies that the length of time the algorithm would take is the square of the number of storage cells multiplied by a cycle rate. Table 1 lists the test times, in seconds, required for algorithms in the order of O(n) and O(nxn) for several device densities. Device cycle times of 100 nsec are assumed.

ARRAY SIZE	ORDER OF ALGORITHM	
	N	N^2
1K	.0001	.105 sec
4K	.0004	1.67 sec
16K	.0016	26.8 sec
64K	.0066	7.16 min
256K	.0260	1.91 hrs
1MEG	.1050	29.1 hrs
4MEG	.4100	500. hrs

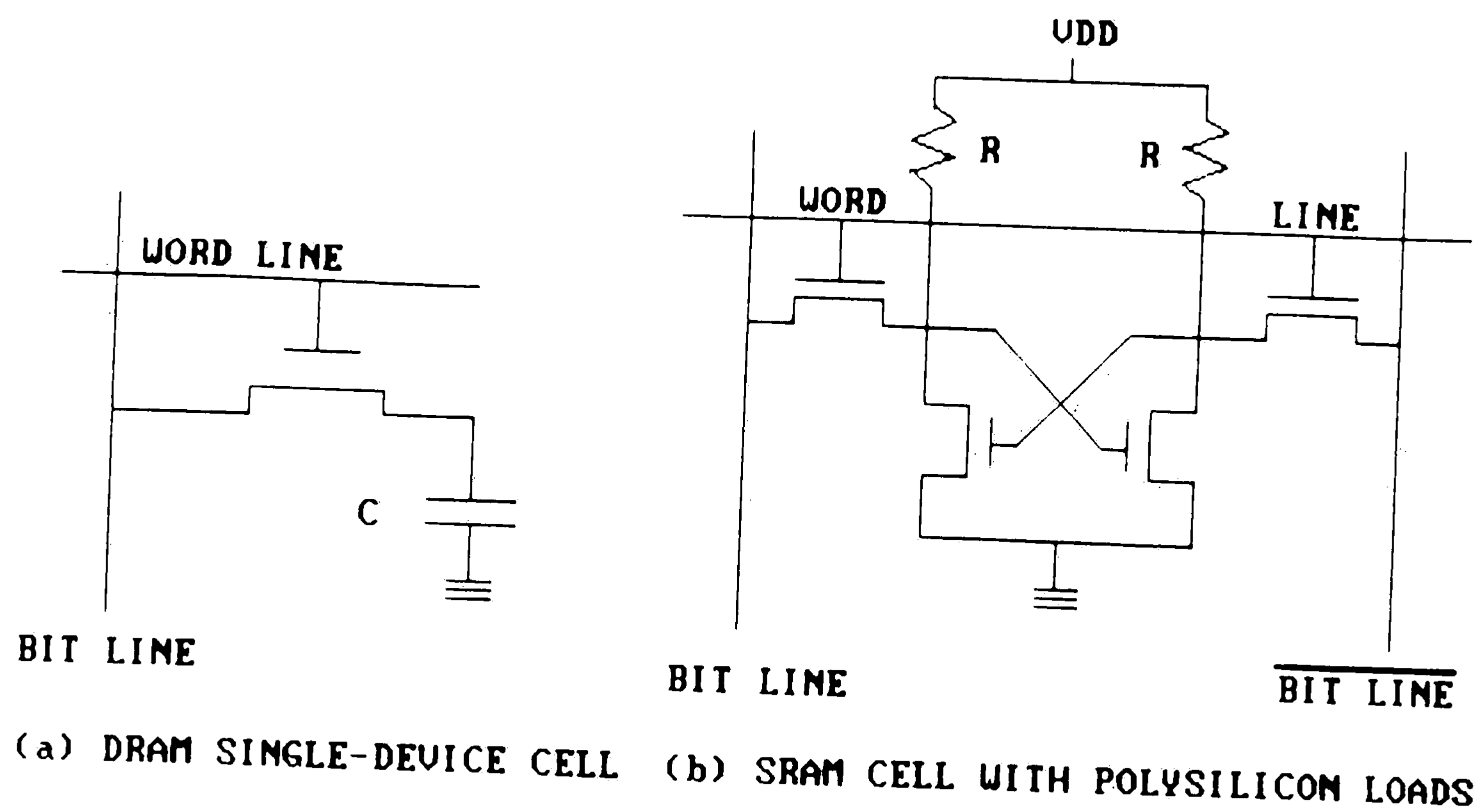
Table 1. Test Times For Algorithms

As the table shows, the density of the device and the type of algorithm used, has a large effect on the test time. Suffice it to say, with the current generation of memories, algorithms in the order of O(n) are a necessity.

An integrated circuit is the result of a multi-step process (~20 mask levels and > 100 processing steps for a 4Mbit DRAM) [Robert Arnold, personal conversation] starting with a silicon wafer and ending with a wafer that has many identical devices with transistors, diodes, capacitors, and resistors interconnected in such a way as to perform a given electrical function. A wafer is processed with many identical die (devices) arranged in rows and columns, separated by saw alleys, allowing the individual die to be separated. (The subject of integrated circuit processing is beyond the scope of this paper). The word "die" will be used to refer to a individual IC memory before the wafer is cut into separate memory chips, i.e. before the memory is separated from the wafer. The words "device" and "memory" will be used interchangeably to represent an individual memory after being separated from the wafer. An integrated circuit (IC) memory therefore is an integrated circuit intended to store information to be retrieved at a later time. IC memories store the information in storage cells as either 1s or 0s. There are three basic types of integrated circuit memories: DRAM, static random access memories (SRAM), and Non-volatile memories. All three types of memories are classified as random access, implying that any cell or

group of cells within the memory may be accessed at any given time and in any order. DRAMs are dynamic random access memories that will only hold the written (stored) information for a specified period of time. (For a 1Mbit DRAM the industry standard is 8 msec). After the storage time has elapsed, the information must be "refreshed" to guarantee that it will not change. The most common storage cell type used in DRAMs are made up of a single transistor and a capacitor (Suk & Reddy, 1980).

SRAMs are static memories, meaning that once information is written into the storage cell, it should remain there until the cell is written to the opposite data. Unlike the DRAMs, SRAMs do not have to be refreshed; however, if powered down, SRAMs will lose the previously stored information. A typical static memory storage cell is made up of 4 transistors and two pull-up resistors (Dekker et al., 1990). Figure 2 shows the most common DRAM and SRAM cell structures.



(a) DRAM SINGLE-DEVICE CELL (b) SRAM CELL WITH POLYSILICON LOADS

Figure 2. DRAM and SRAM Cell Structures

Since it requires more elements to make up a static cell than a dynamic cell, with the same die size more dynamic cells can be processed. Therefore, for a given process technology, denser DRAMs may be manufactured than SRAMs, for approximately the same cost.

Non-volatile memories come in various flavors. They all have the property that once programmed, they retain the information forever. Unlike the SRAMs, Non-volatiles do not lose information when powered down. In general, they are either unable to be programmed by the user or require special hardware to be programmed. The three most common types of non-volatile memories are read only memories (ROMs), erasable programmable read only memories (EPROMs), and electronically erasable

programmable read only memories (EEPROMs). ROMs do not have the capability of being written. The information the ROM contains is processed into the memory during the manufacturing process. EPROMS are programmed by special PROM programming hardware, and are erased by subjecting the device to ultraviolet light. On EEPROMs, information can be erased a word (several bits) at a time electronically, by writing to the chip. These devices have a high power pin on the package that is used for erasing and programming operations.

Recently, specialty memories such as cache-tag, dual-port, and pseudo-statics have found niche applications; however, the bulk of the memories fall within one of the three broad categories. Table 2 gives a world wide perspective of memory usage, in billions of US dollars (McClean, 1990). As you can see, DRAMs have been and will continue to be the predominant device.

	1988	1990	1994
DRAMS	56	54	58
SRAMS	17	22	21
ROMS	8	8	6
EPROMS	17	14	12
EEPROMS	2	2	3
TOTAL	100%	100%	100%

Table 2. Memory Market Share

While the discussions and procedures (to be developed) apply to any type of memory, they are more applicable to the DRAMs and SRAMs.

Appendix A shows an abridged data sheet for a typical 4Mbit DRAM. The first seven pages, pp. 40 to 46, define the DC and AC Electrical Characteristics of the device operation. There are > 50 timing parameters, some with both a minimum and maximum value. The absence of a minimum indicates that the device should operate at or below the stated value. The absence of a maximum value indicates that the device should operate at or above the stated value. Page 40 defines the operating parameters that specify the input voltage levels. Page 41 specifies the operating currents. Please note that there are several modes of operation. All these various parameters and modes of operation must be tested to verify that the device meets the corresponding specifications. Starting on page 47 are three

representative timing diagrams. These are only a sample of the seventeen timing diagrams included in this specification. The Read Cycle timing found on page 47 shows the basic relationships between the various clocks, data, and addresses that are required to perform a read operation. The Early Write Cycle timing on page 48 shows the same relationships as the previous page except for a write operation. Page 49 is an example of how complicated the timing can become on a DRAM. The test engineer must design this timing diagram taking into account all the timing edge relationships shown on this page. In addition to the data sheet requirements, there are memory array related faults and peripheral circuit faults that must be tested to guarantee that the memory is fully functional, i.e. free of these faults.

Array type faults fall within three broad categories: 1) stuck-at, 2) transition, and 3) coupling faults (van de Goor, 1990). Stuck-at faults occur when the stuck-at cell is always in a logical 0 or 1 state, and cannot be changed to the opposite state. A cell that fails to undergo a logical 0->1 transition, or a logical 1->0 transition is classified as a transition fault. A write operation that generates a logical 0->1 or a logical 1->0 transition in one cell and also changes the contents of a second cell is called a

coupling fault. Stuck-at, transition, and coupling faults must be tested to prove cell uniqueness, i.e., that each cell may be written to both logical data types and without any interference from any other cell. DRAMs must be tested for hold time characteristics according to the data sheet. Hold time is the length of time that the DRAM cell can maintain information before needing to be refreshed. Refreshing a cell is accomplished by 1)reading the cell, 2)reading any other cell on the same row, or 3)reading a cell on a row that is internally tied to the target cell's row for refreshing purposes. Single-bit-complement tests are used to verify that the address decoding schemes used by the designer are working properly (Ortner, 1982). Normally, each test in a memory test program is designed to verify one or more of the data sheet parameters, or one of the potential array or circuit faults. As many as two to three hundred individual tests may be necessary to fully test the device.

There are two types of tests that make-up the test program. Pattern generator tests are those that are controlled by a piece of test equipment hardware called the pattern generator. This group of tests looks for errors that occur when the datum read at a particular memory cell is different from the datum expected to be

there. The data sheet timing, cell fault detection tests, cell hold time tests, and address decoding tests are all performed as pattern generator tests. The other type of test is a Parametric Measurement Unit (PMU) test that uses a piece of hardware designed for making very accurate current and voltage measurements. The input and output leakage tests, operating current tests, and output level tests are examples of tests that are made with the PMU. In order to define what a test is composed of, some understanding of the equipment used in memory testing is necessary. In the most general sense, an automated test system is a hardware system that is software controlled. Several of the newer commercial test systems use Sun computers to drive the hardware. C and Pascal are two popular languages for programming these test systems. However, because of the hardware specific commands that are needed, these languages are subsets of the actual languages. Unix has become one of the more popular operating systems for these testers. However, it is the hardware-specific parts of the testers that differentiate one vendor from the other. Although each commercially available tester has the same basic hardware functionality, the accuracy and repeatability of the hardware differentiates the vendors. Before getting to the details of a test, a

quick overview of the different pieces of hardware is necessary. At the heart of each tester is the pattern generator. This piece of hardware controls the addressing algorithms and the data algorithms. Each step in the pattern generator test is defined by as many as 256 bits of information with as many as 256 steps per test. The timing generators can repeatedly place address and clock timing edges, cycle after cycle, with 750 psec accuracy. The comparators are used to detect the output data coming from the device under test. Optional hardware such as Catch Rams and Data Rams are used for storing data that can then be used for comparison with the memory under test or to store the output data from the device to display errors. Current test equipment, configured to test state of the art devices, costs around $1 million, which includes the cost of storage equipment for data collection and analysis.

Basically, a pattern generator test consists of four things: 1)Drive levels, 2)read/write timing, 3)addressing algorithm, and 4)data algorithm. The drive levels specify the power supply voltage during the test, the voltage limits of the address, clock, and data lines, and the expected high and low levels that the data out comparators will use to determine if the output

data are correct. The read/write timing specifies the duration of the test cycles, the time during the test cycle that the addresses will change, and the limits of each of the clock cycles. Also, during a read cycle, the time when the expected data is checked is specified, and during a write cycle, the time that the input data is valid is defined. The addressing algorithm specifies what the addresses are going to do in each cycle, whether it is a read or write cycle, and what clocks are used. The data algorithm defines the expected or input data during each of the address algorithm cycles.

As a new memory is being designed, the test engineer works with the designers to determine the location of the weaknesses of the device and what types of tests should be designed to test for those weaknesses. These specially designed tests, along with the standard data sheet tests, define the development test program. The development test program is designed to test the die while still on the wafer as well as when it is in a package. After the first devices are manufactured, other weaknesses are found and more tests are developed to find all devices with these weaknesses. As more and more devices are produced, a rich set of tests are developed to find all the weaknesses that the devices may have over a range of processing variations.

This test program forms the basis of the production test program.

When a device goes into production, it is not tested just once, but rather goes through several different test points. Figure 3 is a simplified process flow diagram, which shows where each die or package is tested.

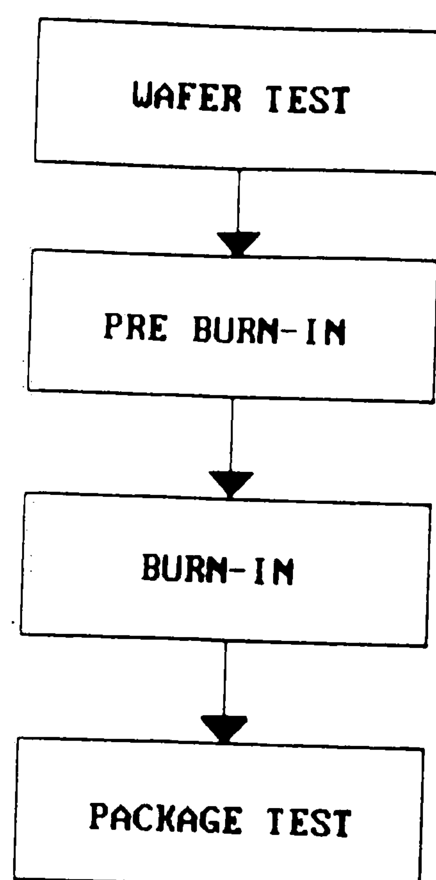


Figure 3. Test Sequence Diagram

First, each die on the wafer is tested and marked if it is good. Depending on the history of the device, and the practices of the manufacturer, not all tests will be performed at the wafer test point. In fact, the actual wafer test program may be a much reduced subset of the package test program. The idea is to maximize package

yield while minimizing test costs. The manufacturer doesn't want to package too many devices that subsequently fail at another test point.

Each good die is separated from the wafer and placed into a package. The pre burn-in test point can be a fairly simple test used to determine if the packaging operation was successful and if the device to be burned-in is still functional. Burn-in is a high temperature (150 degrees C.), high operating bias (7 Volts) test to accelerate early failure mechanisms. Burn-in (usually 24 - 48 hours) is performed by software controlled equipment that is capable of detecting the failures as they occur, so that the devices that fail are removed at that point and do not continue to the final package test point. Test cost considerations and throughput are major factors that determine the time taken on testing devices at all the test points, except final test. The final package test point is the last chance the test engineer has to find all the defective parts.

The remainder of this paper is based on my work and experiences, over the past 10 years at Bell Laboratories, in the area of IC memory testing. Much of the knowledge on memory testing is embodied in relatively few individuals. Activities at Bell

Laboratories, in the area of test time reduction, have been performed by even fewer individuals, and remains an undocumented science. The use of the Summary Database (to be presented later) is commonly used in analyzing the test results of small numbers of devices (< 5,000). Tom Mantz of Bell Laboratories was the first person to use the Summary Database for looking at large numbers of devices (> 1,000,000), specifically to identify tests that fail infrequently. The Signature Format (to be presented later) was developed by Bill Ortner of Bell Laboratories as a format for presenting raw test data. My contributions to test time reduction came in three areas: 1) the use of combinatorial tests, 2) applying the Signature Format to non stop-on-first-fail data along with an algorithm for identifying the minimum set of tests needed to find all failures, and 3) the organization of the Summary Database analysis and the Signature analysis into a automated real time data analysis system for reducing test times.

Conventional Techniques For Reducing Test Times

As a memory device reaches high levels of production, the test engineer should begin to look for ways to reduce the length of time it takes to test the device, hence lowering the overall cost of the finished package. Conventional test time reduction techniques

tend to fall into four categories: reduction of algorithms, test reordering, combinatorial tests, and test removal or relocation. Each of these techniques will be discussed in order, along with the pros and cons of each.

Reduction of Algorithms

The use of algorithms that are in the order of O(n) is a fairly standard practice. However, the use of a 5n algorithm in place of a 17n algorithm on a 4Mbit DRAM would save approximately 5 secs per test. If there were 50 such tests in the test program then 250 sec or over 4 minutes could be saved by the reduction of the algorithm. Unfortunately, most of the work on algorithms tends to support longer algorithms rather than shorter algorithms [van de Goor, 1990]. The number and types of faults that will escape the shorter algorithms is well documented. To the test engineer, the choice is a trade off between the outgoing acceptable quality level (AQL) and the length of the test time. The AQL is a monitored metric throughout the semiconductor industry, and represents the number of test escapes (marginal devices) per 1 million devices tested. For DRAMs the acceptable AQL is between 50 and 100 parts per million (ppm). Therefore, if the test engineer can run experiments over long periods of time

(checking for process variations), and determine that with a given device code and a given process, that the use of shorter algorithms can be substituted for longer algorithms without changing the AQL of the device, then the substitution may safely be done. After the substitution is installed, it is difficult to monitor whether any change in the AQL has been effected. Unfortunately, it is often the customer who finds out that the AQL has changed, with the risk of the customer being lost.

Test Reordering

The second technique used for reducing test times, test reordering, had a greater effect before the introduction of parallel test equipment. To understand the affect test reordering would have on reducing test times, it is useful to understand the testing sequence. In production test programs, the test program stops as soon as a failure occurs. (No sense in spending time testing a part that will not be graded as a shippable device.) There are a few instances when one or more tests may be performed after a failure, but they are special cases intended to collect data to better explain the cause of the failure. All good test programs begin with a test for electrical short circuits between each pin and ground. The test engineer does not want to

power up a device that would apply an electrical short across the test equipment power supply or signal drivers. The next test should be a battery of gross functionality tests that use levels and timing that are looser than those specified by the data sheet. From a processing point of view, it is important to understand the ratio of devices that are grossly functional but not passing, to those devices that are passing. After the loose functional tests, the order of the tests is unimportant except that the tests that check for tight timing should be done last. The tight timing tests are used for sorting the device into various speed categories (speed sorting). Most manufacturers speed sort the device into one or more speed sort categories, so the device should have passed all tests before being graded into a speed sort. Now, suppose that one test in particular leads to more failures than any of the other tests. If this test is late in the sequence, then much time is spent doing tests before reaching the high failure rate test. Therefore, the tests should be ordered in such a way that the highest failure rate tests come first followed by the tests with lower failure rates. However, with the introduction of parallel test equipment, the order of the tests becomes less important. The current state-of-the-art memory

testers (such as the 100 MHz Teradyne J937) are capable of testing 16 devices in parallel. Except for the tests that use the Parametric Measurement Unit (PMU), all tests are run on 16 devices at a time. This means that the tester does not stop testing until all 16 devices either fail or until all the tests are done. With package yields in the high 90% range, the probability of having all sixteen devices fail before all the tests are performed is very remote. Therefore, test order is important except where highly parallel test equipment is used.

Combinatorial Tests

Combinatorial tests, the third technique for test time reduction, has received little attention up to now, but has some interesting points. Although impossible to do, suppose that one test could be designed that would cover all the data sheet requirements as well as the device specific tests. This would probably be a very long test, however, shorter than doing each test individually. This combinatorial technique was tried for combining several production tests that took long periods of time to perform. These combinatorial tests were very difficult, from a software standpoint, to program, and tended to increase the overall failure rates for the device. It is difficult to explain to

management that in order to decrease the overall test time, it is necessary to fail more parts. Test time reduction is important, but not so that it affects the yield. Another draw back to the combinatorial tests is in failure analysis. When each test is designed to find a particular fault in the device, then when that test fails, the reason is evident. However, when several tests are combined into one, and a failure is found, the reason for the failure is not so evident. At this point, the combinatorial test that is failing must be broken into its individual parts to find the failing parameter.

Test Removal or Relocation

The last conventional technique for reducing test times is the most drastic, however, it has the largest impact on the final package test time. Test removal from the final package test point can occur in three different ways. One, the test may simply be eliminated. If the data from all devices tested indicate that the AQL will not be adversely affected by the removal of one or more tests, then that test could be removed. The better approach to test removal from final package test is to move the test into the burn-in test point. The burn-in ovens are capable of doing a limited amount of testing. Since all devices get burned-in for 24 - 48

hours, there is sufficient time for doing long and otherwise expensive tests. The drawback is the limitations of the burn-in systems. The signal accuracy and harsh environment are not conducive to high speed memory testing. However, data hold time tests are not timing dependent, they are very long, and are good candidates for removal to the burn-in test area. Another way to remove tests from the final package test area is to move the tests to the wafer test point. This is a case of robbing Peter to pay Paul. Wafer probing also has high testing costs associated with it. In fact, in most cases, the same model test equipment is used in wafer probe as in final package test. The benefits of moving more tests to wafer probe is that the package yield goes up, i.e., less parts that will fail later ever make it passed this point. The drawbacks are that many things can happen to a device from the time it was tested on the wafer until it is ready for final package testing. It is not wise to assume that a test that passed on the wafer will also pass once the device is in a package.

In the ideal situation, the test engineer would like to have as much unrestricted time as necessary, in final package test, to do an exhaustive group of tests to guarantee an AQL of less than 100 ppm. All of the

conventional test time reduction techniques discussed have in one way or another an adverse affect on the AQL. It is for this reason that I have developed a real time adaptive test time reduction system that will not affect the AQL of the memory device code, yet will allow for minimum test times. I call this technique Adaptive Test Time Reduction.

Adaptive Test Time Reduction

There does not seem to be a generally accepted definition of a self-adaptive control system. However, one definition fits the spirit of use for this thesis, namely: 'A self-adaptive control system is one that provides a means of continuously measuring the system's performance in relation to a given criterion or Figure of Merit, and a means of automatically modifying the system's adjustable parameters by closed loop action so that the Figure of Merit may be satisfied' (Davies, 1970). In applying this definition to a production memory test environment then, the system is required to test memory devices so that an AQL if < 100 ppm. is maintained (Figure of Merit). The system's adjustable parameters are the individual tests that are applied to each device. This system should monitor and analyze the test data as it is being generated, make decisions on whether a test(s) may be eliminated, reorder the

sequence of tests, and then implement the decided changes.

Memory devices travel through the production sequence in a unit called a 'lot'. Initially, the 'lot' is made up of several wafers (< 30) which are processed to contain several individual memory die. The number of die on a wafer is dependent on the size of the wafer (5" or 6") and the size of the individual die (4Mbit DRAMs are physically larger than 1Mbit DRAMs). The number of die per wafer typically ranges between 100 and 300. After wafer testing, the wafers are sawed apart so that the good die may be separated from the bad. At this point the 'lot' is comprised of many individual die which are bonded into one of various types of packages (the finished state). These packages now make up the 'lot'. The size of the 'lot' may vary from several hundred to several thousands of devices. The size of the 'lot' is unimportant, but the concept that the memories travel in distinct units ('lots') is.

When a 'lot' of devices arrives to be tested, among the information traveling with the 'lot' are the 'lot' identifier, and the total number of devices that comprise the 'lot'. Before testing begins, the operator of the automated test equipment (ATE), enters the lot identifier into the software controlling the ATE.

Before the test software performs the sequence of tests on the device, two files are opened on the data collection hardware, which are called concentrators, with the 'lot' identifier as the first part of the file name, and one file with a .dat extension and another file with a .sum extension. As each test is performed on the device, the test number and test result are sent to the .dat file. After all the devices in the 'lot' are tested, the ATE closes the .dat file and sends a summary of failing tests to the .sum file. For each test that failed, the .sum file includes the test number, the number of times that it was run, and the number of devices that failed it. Then the .sum file is closed. At this point, two files exist on the concentrator that contain all the needed information for the final disposition of the packaged devices. The .dat file has the raw test data for each device in the 'lot' and the .sum file has the statistics for the lot as a whole. It is at this point where the conventional procedures for memory testing end. The good packages are separated from the bad, with the good packages being ready for shipment. The test data has been collected and stored, but only if unusual test results or yields have been found, are the data given much attention.

The files that are created during the testing

process have been identified; however, the hardware requirements are dependent on the size of the test center. Since a generic approach to test time reduction is being presented, the capabilities of the hardware would have to be evaluated based on the particular hardware utilized. Figure 4 shows a typical hardware configuration for a multi-tester test environment.

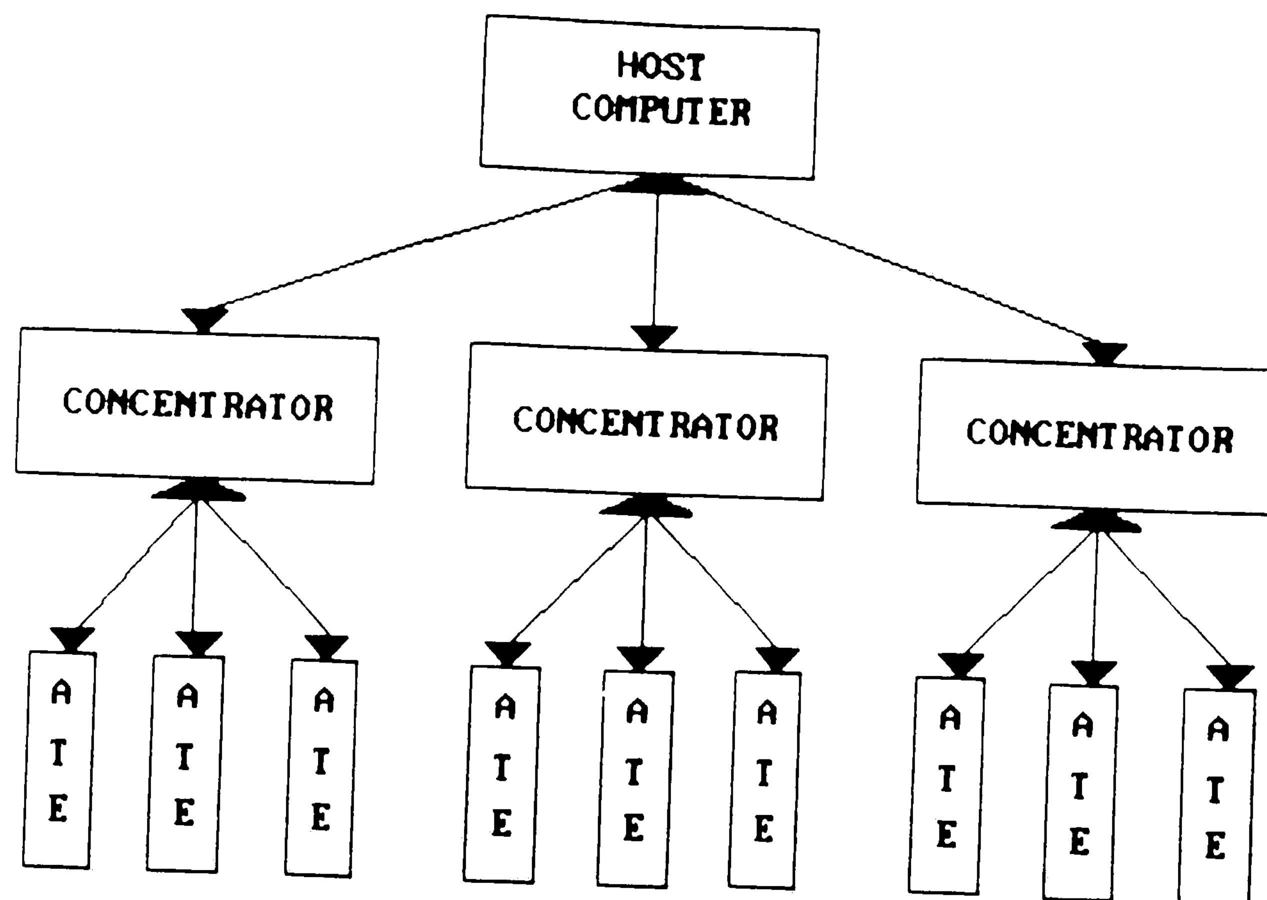


Figure 4. Multi-Tester Production Environment

For each group of testers(ATE), a concentrator is recommended. The concentrator(s) are then networked to a single host computer.

It is the job of the host computer to perform the long term storage of all the test data. Therefore, the host computer must have large hard disk storage, along

with some form of tape media for archival purposes. The data being generated by the ATE and sent to the concentrators are transferred to the host computer by the concentrator(s). The host computer maintains a chronological table of the XXX.sum and XXX.dat files as it receives the files, runs the software that analyzes the data, and controls the self-adaptive system.

The following self-adapting testing environment uses a two fold approach. First, summary analysis (SA) is used to identify tests whose rate of failure is below the accepted AQL, and second, failure signature analysis (FSA) is used to 1) identify tests that fail uniquely and 2) identify test coverages. Each of the two approaches will be explained in turn.

Figure 5 shows a truncated summary sheet composed of devices from several 'lots'. Assuming the tests were performed in the order listed, each successive test was performed fewer times than the one before, because of the stop-on-first-fail nature of the testing. Stop-on-first-fail means: as soon as a device fails a test, no further tests are performed on that device. The important information from this database is the failure rate column. Assuming a desired AQL of < 100 ppm (failure rate of .0001), then by eliminating tests 22, 2, and 303, 88 defective parts would have escaped

testing; however, the AQL would still be < 100 ppm. The software analyzing these data identifies as many tests as possible for elimination (hence reducing test time by the greatest amount). Therefore, by selecting the tests in the order of smallest failure rate first, the largest number of tests will be identified. There is one problem with this test elimination approach. Once the identified tests are removed, they no longer appear in the data base, and therefore no longer contribute to the AQL. At this point, no further test removal is possible because all the AQL allowed was used-up by these tests. With the passage of time, these previously removed tests may have changed their failure rates either up or down. Therefore, it is very important for the ongoing success of the adaptive system to sample the removed tests to maintain a database of the failure rates for these tests.

Test#	#Times Run	#Failures	Failure Rate
1	1,087,536	510	.0004689
2	1,087,026	23	.0000212
3	1,087,003	2387	.0021959
22	1,084,616	9	.0000083
23	1,084,607	187	.0001724
24	1,084,420	5612	.0051751
31	1,078,808	927	.0008593
32	1,077,881	82	.0000761
33	1,077,799	447	.0004147
300	1,077,352	1783	.0016549
301	1,075,569	365	.0003394
302	1,075,204	691	.0006427
303	1,074,513	56	.0000521
.	.	.	.
.	.	.	.
.	.	.	.
359	1,017965	8327	.0081800

Figure 5. Summary Database

Since this is an adaptive system, the vintage of the data is important. As the 'lots' are tested, by possibly many different testers, the database is constantly changing. As each new summary is sent, the summation data being used for identifying test removal should be updated. Using a FIFO method, the latest 1 million devices tested should be used as the basis for the test removal analysis. This is easily accomplished by summing all the .sum files, starting with the most recently closed file, until greater than 1 million devices comprise the database. At this point, the analysis software should be run to identify whether any changes should be made to the tests that have been eliminated.

The second approach to the adaptive system is the use of failure signature analysis (FSA) to 1) identify tests that fail uniquely, and 2) to identify test coverages. A signature is the pattern of tests that a device passes and fails during testing. FSA is the process of analyzing the pass and fail signatures exhibited by a group of devices when stop-on-first-fail testing is not used. As mentioned previously, production testing is performed stop-on-first-fail. To generate the data needed for the signature analysis, periodically all tests must be performed on the devices. The test software would need to be written so that all tests could be sampled at a predetermined rate, while maintaining the reject criteria set-up in the stop-on-first-fail strategy. A test that fails uniquely is one that, for any given device failure, is the only test that the device fails, based on all tests being run. Test coverage is the case where a group of devices fail more than one test, but each device in the group fails at least one test in-common with all the other devices. Figure 6 is an example of failure signatures (.dat file). The first column is the frequency, or number of devices that failed the exact same tests. The numbers over the table are the test numbers, a '*' means the test passed, and a '.' means the test failed.

Figure 6. Signature Analysis

A signature analysis can be performed on any number of devices; however, the larger the number of devices, the more interesting the result. One can learn from Figure 6 exactly the minimum set of tests required to find all the defective devices in this population. By reordering the tests, so that this minimum set of tests is performed first, theoretically these would be the only tests required to find all defective devices. Any tests in the test sequence beyond these tests would have failure rates so low that many, if not all, would be eliminated by the failure rate analysis. To find the minimum set of tests needed to find all the defective

devices, first identify any signature that has only one failure (a unique test failure). Obviously, to catch this failure, the test at the top of the column must be done. Next, all signatures that have this test as one of its failing tests can be ignored (if this were the only test performed, then these devices would have failed based on this one test). Then identify the next signature with the least number of failing tests. Pick the test, out of the ones that fail, that would eliminate the most signatures out of the remaining signatures. Mark this test as must do and ignore all signatures that have this test in their signature. Repeat this process until every signature has been accounted for. Referring to Figure 6, one device failed test 342 only, therefore this test must be performed. If test 342 were the only test done, then the devices that had the signatures with frequency 9341, 5094, 3745, 934, 596, and 230 would also have been rejected. The next signature has three failing tests. By selecting test 358, this device would have been rejected, and the devices with signature frequencies of 104, 90, 88, 76, 55, 32, 19, and 14 would also have been rejected. The next signature, with a frequency of 10, has eight failing tests. It doesn't matter which test is selected, since no other remaining signatures fail any

of these tests. Therefore, select test 301. There are now only two signatures left. The next highest signature, with a frequency of 27, failed ten tests. By selecting test 307, all devices with this failure signature as well as all those with the signature of frequency 31 would be rejected. By only performing tests 342, 358, 301, and 307 all the devices in this population would have been rejected. Therefore, if these tests were the first four tests performed, then in the stop on first fail database, all the other tests would have failure rates of .00000 and would qualify for elimination.

The host computer would perform the summary analysis and signature analysis each time a new 'lot' of devices was added to the database. The results of the analysis (tests removed, tests added, or test reordering) would be sent down to the concentrators in the form of a table of test numbers in the order of execution, and whether the test must be sampled or not.

The concentrator(s) are the interface between the ATE and the host computer. Since all the 'lot' data must be analyzed with all the other 'lots', a single computer (host) must handle this analysis. However, with multiple ATE generating data simultaneously, enough concentrators must be available to handle this

continuous flow of data, so that no data are lost, and the ATE do not have to wait until the concentrator is ready to receive its data. In addition to collecting the data from several ATE, and passing those data to the host computer, the concentrator transfers the table of test numbers to be performed to the ATE.

The ATE must use a procedural test language that is capable of defining each test uniquely, and be able to read a file with a list of test numbers to be performed and then execute that list. Figure 7 shows a typical unique procedure that would perform a single test.

```
PROCEDURE TEST301
     CTN 301            /* set test number              */
     CALL LEVELS_1      /* clock and address levels     */
     CALL TIMING_1      /* timing conditions            */
     CALL PATTERN_1     /* address and data algorithms  */
     RUN_TEST           /* run the test                 */
END
```

Figure 7. Typical Test Procedure

In order to eliminate the need for operator intervention, the test software running on the ATE would need to maintain two test tables. The first test table would simply include a list of all the tests that the test program was designed to run. This table would be used if the initial attempt by the test program to read the test table from the concentrator fails. The second test table, or "working table", is the table that was obtained from the concentrator. The working table

controls the sequence of the testing performed by the ATE. This table contains a list of test numbers that should be executed by the ATE, along with a number (N) that represents 1 out of N times to run the test (sampling rate). A '0' would specify a test that must always be run, a '1' would specify a test that currently must be run 100% of the time, but could be a candidate for sampling, and every sampled test could have a different number N such that the ability to sample at different rates was possible. The working table may be modified by the ATE for individual 'lots'. The ATE may modify the table by upgrading any sampled test to being run 100% of the time on 'lots' that experience a failure of a sampled test, however, the tests that are run 100% of the time cannot be downgraded to sampling status by the ATE. As the test program scans the working table, it first does all those tests that have a '0' or '1' for its sampling rate. Then a second pass is done looking for tests that have a number other than '0' or '1'. If a test is identified as a sample test (it's sampling rate is not a '0' or a '1'), then a random number is generated between 1 and N. If a '1' is generated, then the test is executed, if a number other than a '1' is generated, then the test is not run on this device.

To initialize the system, simply conduct full

testing on 1,000,000 devices and start the analysis software running on the host computer. After a new test table is transferred from the host to the concentrator(s), and as each ATE finishes a 'lot' and checks the concentrator for an updated test table, the full system will begin to function.

Given a memory device code that is running at a significant level of production, optimal testing with minimum test times could be achieved with an adaptive test program approach. The hardware that would be employed is generally available, and the software would not be too difficult to design. Time and experience with the system would be necessary to fine tune the sampling rates so that the overall AQL for the memory device code in production would not be negatively affected.

References

Abadir, M.S. and J.K. Reghbati (1983). Functional Testing of Semiconductor Random Access Memories. A.C.M. Computing Surveys, pp. 175-198.

Aloneftis,A. (1987). Lecture Notes in Control and Information Sciences. Springer-Verlag: Berlin Heidelberg New York.

Brassard, G. and P. Bratley (1988). Algorithmics Theory and Practice. Prentice Hall. New Jersey.

Canudas De Wit, C.A. (1988). Adaptive Control for Partially Known Systems: Theory & Applications. Elsevier, The Netherlands.

Davies, W.D.T. (1970) System Identification for Self-Adaptive Control. John Wiley & Sons Ltd. U.K.

Dekker, R. et al. (1988). Fault Modeling and Test Algorithms for Static Random Access Memories. In Proc. Int. Test Conference, Washington D.C.,pp. 343-352.

Dekker, R. et al. (1990). A Realistic Fault Model and Test Algorithms for Static Random Access Memories. IEEE Trans. on Computers, pp. 567-572.

van de Goor, A.J. and C.A. Verruijt (1990). An Overview of Deterministic Functional RAM Chip Testing. ACM Computing Surveys, pp. 5-33.

van de Goor, A.J. (1990). State of the Art Memory Testing From Chips to Boards. John Wiley & Sons. The Netherlands.

Hayes, J.P. (1975). Detection of Pattern-Sensitive Faults in Random Access Memories. IEEE Trans. on Computers, pp. 150-157.

Hayes, J.P. (1980). Testing Memories for Single-Cell Pattern-Sensitive Faults. IEEE Trans. on Computers, pp. 249-254.

de Jonge, J.H. and A.J. Smeulders (1976). Moving Inversions Test Pattern is Thorough, Yet Speedy. Computer Design, pp. 169-173.

de Jong, P. and A.J. van de Goor (1988). Comments on 'Test Pattern Generation for API Faults in RAM'. IEEE Trans. on Computers, pp. 1426-1428.

Knaizuk Jr., J. and C.R.P. Hartman (1977). An Optimal Algorithm for Testing Stuck-at Faults in Random Access Memories. IEEE Trans. on Computers, pp. 1141-1144.

Lewyn, L.L. and J.D. Meindl (1985). Physical Limits of VLSI DRAMs. IEEE Journal of Solid State Circuits, pp. 231-241.

Marinescu, M. (1982). Simple and Efficient Algorithms for Functional RAM Testing. In Proc. IEEE Int. Test Conference, pp. 236-239.

Mazumder, P. and J.H. Patel (1988). Parallel Testing of Parametric Faults in a DRAM. In Proc. of the fifth MIT Conference. Cambridge, MA., pp. 149-161.

McClean, W.J. (1990). Status 1990, A Report on the Integrated Circuit Industry. Integrated Circuit Engineering Corporation: Arizona, U.S.A.

Nair, R., S.M. Thatte and J.A. Abraham (1978). Efficient Algorithms for Testing Semiconductor Random Access Memories. IEEE Trans. on Computers, pp. 572-576.

Nair. R. (1979). Comments on 'An Optimal Algorithm for Testing Stuck-at Faults in Random Access Memories'. IEEE Trans. on Computers, pp. 258-261.

Ortner, W.R. (1982). The Methodology and Concepts of Memory Testing - Emphasis on STATIC RAMS. Bell Laboratories Technical Memorandum.

Papachristou, C.A. and N.B. Saghal (1985). An Improved Method for Detecting Functional Faults in Random Access Memories. IEEE Trans. on Computers, pp. 110-116.

Saluja, K.K. and K. Kinoshita (1985). Test Pattern Generation for API Faults in RAM. IEEE Trans. on Computers, pp. 284-287.

Suk, D.S. and S.M. Reddy (1980). Test Procedures for a Class of Pattern Sensitive Faults in Semiconductor Random Access Memories. IEEE Trans. on Computers, pp. 419-429.

Suk, D.S. and S.M. Reddy (1981). A March Test for Functional Faults in Semiconductor Random Access Memories. IEEE Trans. on Computers, pp. 982-985.

Thatte, S.M. and J.A. Abraham (1977). Testing of Semiconductor Random Access Memories. In Proc. of the 7th Annual Int. Conference on Fault-Tolerant Computing, pp. 81-87.

Winegarden, S. and D. Pannell (1981). Paragons for Memory Test. In Proc. IEEE Int. Test Conference, pp. 4-48.

APPENDIX A

Parameter	Symbol	Value	Unit
Voltage on any pin relative to V_{SS}	V_T	-1.0 to +7.0	V
Supply voltage relative to V_{SS}	V_{CC}	-1.0 to +7.0	V
Short circuit output current	I_{out}	50	mA
Power dissipation	P_T	1.0	W
Operating temperature	T_{opr}	0 to +70	°C
Storage temperature	T_{stg}	-55 to +125	°C

Electrical Characteristics

* Recommended DC Operating Conditions (Ta = 0 to +70 °C)

Parameter		Symbol	Min	Typ	Max	Unit	Note
Supply voltage		V_{SS}	0	0	0	V	
		V_{CC}	4.5	5.0	5.5	V	1
Input high voltage		V_{IH}	2.4	-	6.5	V	1
Input low voltage	(I/O pin)	V_{IL}	-1.0	-	0.8	V	1
	(Others)	V_{IL}	-2.0	-	0.8	V	1

Note: 1. All voltage referenced to V_{SS}

* DC Electrical Characteristics (Ta = 0 to +70 °C , V_{CC} = 5 V ± 10 %, V_{SS} = 0 V

Parameter	Symbol	HM514400 -8 Min	HM514400 -8 Max	HM514400 -10 Min	HM514400 -10 Max	HM514400 -12 Min	HM514400 -12 Max	Unit	Test Conditions
Operating current	I_{CC1}	-	90	-	80	-	70	mA	$\overline{RAS}$,$\overline{CAS}$ cycling t_{RC} = min
Standby current	I_{CC2}	-	2	-	2	-	2	mA	TTL interface $\overline{RAS}$,$\overline{CAS}$=V_{IH} DOUT=High-Z
		-	1	-	1	-	1	mA	CMOS interface $\overline{RAS}$,$\overline{CAS}$≥V_{CC}-0.2V DOUT=High-Z
$\overline{RAS}$-only refresh current	I_{CC3}	-	90	-	80	-	70	mA	t_{RC} = min
Standby current	I_{CC5}	-	5	-	5	-	5	mA	$\overline{RAS}$ = V_{IH} $\overline{CAS}$ = V_{IL} Dout = enable
$\overline{CAS}$-before-$\overline{RAS}$ refresh current	I_{CC6}	-	90	-	80	-	70	mA	t_{RC} = min
Fast page mode current	I_{CC7}	-	90	-	80	-	70	mA	t_{PC} = min
Input leakage current	I_{LI}	-10	10	-10	10	-10	10	μA	0 V ≤ V_{IN} ≤ 7 V
Output leakage current	I_{LO}	-10	10	-10	10	-10	10	μA	0 V ≤ V_{OUT} ≤ 7 D_{OUT} = disable
Output high voltage	V_{OH}	2.4	V_{CC}	2.4	V_{CC}	2.4	V_{CC}	V	High I_{OUT} = -5
Output low voltage	V_{OL}	0	0.4	0	0.4	0	0.4	V	Low I_{OUT} = 4.2

Note: 1. I_{CC} depends on output load condition when the device is selected, I_{CC} max is specified at the output open condition.

2. Address can be changed once or less while $\overline{RAS}$ = V_{IL}.

3. Address can be changed once or less while $\overline{CAS}$ = V_{IH}.

* Capacitance (T_a = 25 °C, V_{CC} = 5 V ± 10 %)

Parameter	Symbol	Typ	Max	Unit	Note
Input capacitance (Address)	C_{I1}	-	5	pF	1
Input capacitance (Clocks)	C_{I2}	-	7	pF	1
Output capacitance (Data-in, Data-out)	$C_{I/O}$	-	10	pF	1,2

Note: 1. Capacitance measured with Boonton Meter or effective capacitance measuring method.
2. $\overline{CAS}$ = V_{IH} to disable Dout.

* AC Characteristics (T_a = 0 °C to 70 °C, V_{CC} = 5 V ± 10 %, V_{SS} = 0 V) 1),

Read, write, read-modify-write and refresh cycles (Common parameters)

Parameter	Symbol	HM514400 -8		HM514400 -10		HM514400 -12	
		Min	Max	Min	Max	Min	Max
Random read or write cycle time	t_{RC}	150	---	180	---	210	---
$\overline{RAS}$ precharge time	t_{RP}	60	---	70	---	80	---
$\overline{RAS}$ pulse width	t_{RAS}	80	10000	100	10000	120	10000
$\overline{CAS}$ pulse width	t_{CAS}	25	10000	25	10000	30	10000
Row address set-up time	t_{ASR}	0	---	0	---	0	---
Row address hold time	t_{RAH}	12	---	15	---	15	---
Column address set-up time	t_{ASC}	0	---	0	---	0	---
Column address hold time	t_{CAH}	15	---	20	---	25	---
$\overline{RAS}$ to $\overline{CAS}$ delay time	t_{RCD}	22	55	25	75	25	90
$\overline{RAS}$ to column address delay time	t_{RAD}	17	40	20	55	20	65
$\overline{RAS}$ hold time	t_{RSH}	25	---	25	---	30	---
$\overline{CAS}$ hold time	t_{CSH}	80	---	100	---	120	---
$\overline{CAS}$ to $\overline{RAS}$ precharge time	t_{CRP}	5	---	10	---	10	---

Read, write, read-modify-write and refresh cycles (Common parameters)

Parameter	Symbol	HM514400 -8 Min	HM514400 -8 Max	HM514400 -10 Min	HM514400 -10 Max	HM514400 -12 Min	HM514400 -12 Max
$\overline{OE}$ to D_{IN} delay time	t_{ODD}	20	---	25	---	30	---
$\overline{OE}$ delay time from D_{IN}	t_{DZO}	0	---	0	---	0	---
$\overline{CAS}$ set-up time from D_{IN}	t_{DZC}	0	---	0	---	0	---
Transition time (rise and fall)	t_T	3	50	3	50	3	50
Refresh period	t_{REF}	---	16	---	16	---	16

Read cycle

Parameter	Symbol	HM514400 -8 Min	HM514400 -8 Max	HM514400 -10 Min	HM514400 -10 Max	HM514400 -12 Min	HM514400 -12 Max
Access time from $\overline{RAS}$	t_{RAC}	---	80	---	100	---	120
Access time from $\overline{CAS}$	t_{CAC}	---	25	---	25	---	30
Access time from address	t_{AA}	---	40	---	45	---	55
Access time from $\overline{OE}$	t_{OAC}	---	25	---	25	---	30
Read command set-up time	t_{RCS}	0	---	0	---	0	---
Read command hold time to $\overline{CAS}$	t_{RCH}	0	---	0	---	0	---
Read command hold time to $\overline{RAS}$	t_{RRH}	10	---	10	---	10	---
Column address to $\overline{RAS}$ lead time	t_{RAL}	40	---	45	---	55	---
Output buffer turn-off time	t_{OFF1}	0	20	0	25	0	30
Output buffer turn-off to $\overline{OE}$	t_{OFF2}	0	20	0	25	0	30
$\overline{CAS}$ to D_{IN} delay time	t_{CDD}	20	---	25	---	30	---

Write cycle

Parameter	Symbol	HM514400 -8 Min	HM514400 -8 Max	HM514400 -10 Min	HM514400 -10 Max	HM514400 -12 Min	HM514400 -12 Max
Write command set-up time	t_{WCS}	0	---	0	---	0	---
Write command hold time	t_{WCH}	15	---	20	---	25	---
Write command pulse width	t_{WP}	15	---	20	---	25	---
Write command to $\overline{RAS}$ lead time	t_{RWL}	25	---	25	---	30	---
Write command to $\overline{CAS}$ lead time	t_{CWL}	25	---	25	---	30	---
Data-in set-up time	t_{DS}	0	---	0	---	0	---
Data-in hold time	t_{DH}	15	---	20	---	25	---

Read-modify-write cycle

Parameter	Symbol	HM514400 -8 Min	HM514400 -8 Max	HM514400 -10 Min	HM514400 -10 Max	HM514400 -12 Min	HM514400 -12 Max
Read-modify-write cycle time	t_{RWC}	210	---	245	---	285	---
$\overline{RAS}$ to $\overline{WE}$ delay time	t_{RWD}	110	---	135	---	160	---
$\overline{CAS}$ to $\overline{WE}$ delay time	t_{CWD}	55	---	60	---	70	---
Column address to $\overline{WE}$ delay time	t_{AWD}	70	---	80	---	95	---
$\overline{OE}$ hold time from $\overline{WE}$	t_{OEH}	25	---	25	---	30	---

Refresh cycle

Parameter	Symbol	HM514400 -8 Min	HM514400 -8 Max	HM514400 -10 Min	HM514400 -10 Max	HM514400 -12 Min	HM514400 -12 Max
$\overline{CAS}$ set-up time ($\overline{CAS}$-before-$\overline{RAS}$ refresh cycle)	t_{CSR}	10	---	10	---	10	---
$\overline{CAS}$ hold time ($\overline{CAS}$-before-$\overline{RAS}$ refresh cycle)	t_{CHR}	20	---	20	---	25	---
$\overline{RAS}$ precharge to $\overline{CAS}$ hold time	t_{RPC}	10	---	10	---	10	---

Fast page mode cycle

Parameter	Symbol	HM514400 -8 Min	HM514400 -8 Max	HM514400 -10 Min	HM514400 -10 Max	HM514400 -12 Min	HM514400 -12 Max
Fast page mode cycle time	t_{PC}	55	---	55	---	65	---
Fast page mode $\overline{CAS}$ precharge time	t_{CP}	10	---	10	---	15	---
Fast page mode $\overline{RAS}$ pulse width	t_{RASC}	---	100000	---	100000	---	100000
Access time from $\overline{CAS}$ precharge	t_{ACP}	---	50	---	50	---	60
$\overline{RAS}$ hold time from $\overline{CAS}$ precharge	t_{RHCP}	50	---	50	---	60	---
Fast page mode read-mofify -write cycle time	t_{PCM}	105	---	110	---	130	---

Test mode cycle

Parameter	Symbol	HM514400 -8		HM514400 -10		HM514400 -12	
		Min	Max	Min	Max	Min	Max
Test mode $\overline{WE}$ set-up time	t_{WS}	0	---	0	---	0	---
Test mode $\overline{WE}$ hold time	t_{WH}	20	---	20	---	20	---

Counter Test Cycle

Parameter	Symbol	HM51440C -8		HM514400 -10		HM514400 -12	
		Min	Max	Min	Max	Min	Max
$\overline{CAS}$ precharge time in counter test cycle	t_{CPT}	40	---	50	---	60	---

TIMING WAVEFORM

(1) Read Cycle

• ▨ : Don't care

(2) Early Write Cycle

t_{RC}
$\overline{RAS}$
t_{RAS}
t_{RP}
t_T
t_{RSH}
t_{RCD}
t_{CAS}
t_{CRP}
t_{CSH}
$\overline{CAS}$
t_{ASR}
t_{RAH}
t_{ASC}
Address
Row
Column
t_{WCS}
t_{CAH}
t_{WCH}
$\overline{WE}$
t_{DS}
t_{DH}
Din
Din
Dout
HI-Z

- $\overline{OE}$: Don't care
- ▨ : Don't care

(10) Fast Page Delayed Write Cycle

RAS

CAS

Address

WE

Din

Dout

OE

tRASC tRP tCRP tRSH tCAS tCSH tPC tT tRCD tCP tRAH tCAH tASR tASC Row Col. tRCS tCWL tRWL tWP tDH tDS Din tOEH Hi-Z tODD

• ▨ : Don't care

Biography

Glen A. Smith has worked for AT&T Bell Laboratories since 1977. Currently a Member of Technical Staff, he is working in a group that qualifies and characterizes general trade integrated circuit memories for the use in AT&T systems. He recieved a BS degree in Commerce from Rider College in 1973 and a BS degree in EET from Spring Garden College in 1976. Born in Walnutport Pennsylvania, in 1951, he now resides in New Tripoli Pennsylvania, along with his wife and three children.