An Analysis of an Inexpensive Memory Test Solution

Ryan Pennucci, Ryan Jurasek, Wolfgang Hokenmaier, Lester Patrick, Jacob Bucci, Donald Labrecque and David Kinney
Green Mountain Semiconductor Inc.
Burlington, VT 05401
Email: rpennucci@greenmountainsemi.com, rjurasek@greenmountainsemi.com

***Abstract*—Multi-project wafers have lowered manufacturing costs for semiconductor prototypes, yet test costs remain high, presenting a barrier for innovation in the market. We present and analyze a low-cost test strategy for memory devices.**

***Index Terms*—low cost, memory**

I. Introduction

Historically, startup semiconductor design companies have been hampered by exorbitant cost barriers to enter into the market. Both manufacturing and testing require significant investment into specialized facilities and equipment. With the emergence of independent foundries, prototype production costs have been significantly lowered. In particular, multi-project wafer services allow multiple customers to share the costs of production by combining several dies for low-volume parts onto each wafer. While this lowers the cost of production, testing remains expensive.

In addition to the high cost of conventional test equipment, specialized facility features are typically required, including raised flooring, high voltage, pneumatic supplies and specialized cooling. In recent years, desktop test equipment has reduced both the cost of equipment and required facilities [1]. These systems offer much of the performance of their full-size counterparts, with lower cost and fewer required facilities. Today, these systems can be rented or purchased used to further reduce equipment costs. Nevertheless, equipment costs may be significantly greater than manufacturing costs.

Beyond the high direct costs of equipment needed for testing, investment is needed to develop test protocols. Generation of test patterns for execution is a complex and time-consuming process which requires an understanding of the device and its failure modes to develop effective tests. Designing a built-in self test (BIST) is still more complex, since a fault model must be developed for the device before a circuit can be designed to search for faults. In memory production, testing is further complicated by the need for test algorithms to locate defects and repair them using redundant features.

Although automatic testing is required for production at scale, it may be an unnecessary burden for research or the early stages of design verification. For such projects, the ability to easily design experiments to characterize the device and to rapidly iterate on these experiments is more valuable. It may be more effective to design a general-purpose BIST which works with the available test equipment to execute tests and to manually manage defect repairs.

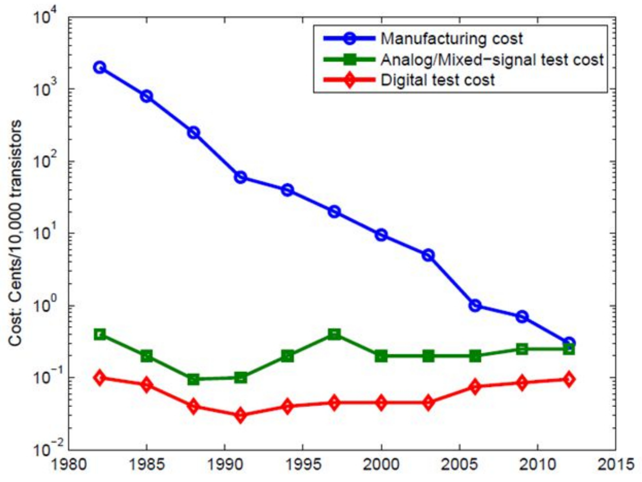

Fig. 1. Semiconductor Manufacturing vs Test Cost [2]

In this paper, we present a low-cost solution to test experimental semiconductor memory designs, consisting of a combination of circuit design elements, test equipment and software features. Many of the techniques shown here are not limited to memory testing, and may be adapted for semiconductor prototype development in general. This solution is configurable to allow for changing interface specifications and design characteristics, while remaining simple and easy to use. Many of the design elements are intended to have value even after a more expensive high volume manufacturing solution is implemented by reducing the resources needed to test each chip.

A. Motivation

As a small company, Green Mountain Semiconductor, Inc. (GMSi) has a limited testing budget. Although memory test equipment is very powerful, our limited requirements for a recent research project along with the high cost of such equipment called for a different approach. To work within our budget while still meeting our needs, we developed a test environment using general purpose instruments. This kept the number of non-reusable components such as specialized test fixtures to a minimum, while allowing adaptation to other products and interfaces. Features that these instruments were unable to provide were instead built into the prototype device under test (DUT) itself.

B. Conventional Testing

Conventional memory test systems are complex and expensive pieces of equipment [3]. They are designed to execute test programs which can analyze all aspects of AC and DC circuit operation which are required from the first silicon design verification through production testing of wafers and packaged devices. In order to operate at sufficiently high speeds, these systems typically execute compiled programs to generate vectors, although precomputed vectors may also be used.

These vectors access the rows and columns of very large memory arrays, often across several devices in parallel. Comparators in the tester are programmed with expect data and used to validate the behavior of each address. A map of passing and failing addresses within the array is kept in storage known as Catch-RAM. These systems often include redundancy repair algorithms to allow spare elements to replace failing portions of the array in order to increase the percentage of fault-free devices. They are also used to obtain failure counts and bit fail maps showing the distribution of failing addresses.

Testing large memory arrays takes a significant amount of time. This test time can be effectively managed through high device parallelism and total automation. Systems are typically capable of a large number of complex memory patterns [4] to be executed with minimal overhead in loading and unloading software into the executable portion of the code.

While these systems are very flexible and can be configured to test many different types of memory devices, the price of new systems typically starts at several hundred thousand dollars, and often runs into the several million dollar range. In addition to the equipment itself, the installation demands costly resources for electricity, clean pressurized air, vacuum and sometimes liquid cooling or liquid nitrogen for the high speed electronics, in order to maintain internal tester temperature and humidity, as well as for parts and wafer handling. Used systems also frequently face high maintenance costs for diagnosis and replacement of defective system components. Desktop test systems are available at lower cost and with fewer required facilities, but these systems are still costly. These cost and facility constraints are often a serious impediment for a small company to overcome.

Beyond the cost of the systems and facilities themselves, effort must be invested to produce test programs to evaluate device performance. More complex devices may additionally integrate a BIST to automate testing. In production-ready devices, robust test programs or BIST circuits are used to identify and correct defects using redundant structures. The design of these tests is complex and requires substantial development effort.

For research projects or design verification, test programs of this complexity are less important. A wider range of parameters must be explored, necessitating many test programs. The complexity of creating test programs makes it difficult to quickly iterate on these programs.

C. GMSi's Approach

The GMSi research project was designed to be tested without using a traditional memory tester. Instead, a large amount of functionality was built into the chip itself and configured during testing. Because the chip integrates many of the functions of a memory tester, low-cost, general-purpose test equipment is able to take the place of expensive, dedicated memory test equipment. A pattern generator supplies vector inputs, while a logic analyzer records inputs and outputs. Paired with a programmable waveform generator acting as a clock, programmable power supplies, and additional instruments, a wide range of operating conditions can be produced to explore many device characteristics.

All testing is performed against a Verilog model of the DUT. The model is wrapped in a behavioral testbench to supply its inputs. By capturing these inputs to serve as stimulus for testing against hardware, a single testbench is able to serve multiple purposes. When designing the test, the internal signals of the model can be inspected. This information is helpful both while designing the test and while diagnosing failures. In addition, the testbench produces output in a customized format which allows it to call out to the software controlling the test instruments. The testbench is able to request that operating conditions be changed or specific measurements be performed.

II. Testing

A. Hardware Design

The purpose of GMSi's effort was to explore a DRAM design that incorporates additional functionality into the memory device itself. This functionality requires modifications to the signal paths between the memory's primary and secondary sense amplifiers but does not depend on the memory array itself. To simplify the prototype, the DRAM array was removed, leaving the primary sense amplifiers intact. Data is read and written on the sense amplifiers themselves, allowing the remainder of the design to be tested while reducing the cost, size and complexity of the prototype significantly.

Without a DRAM array, there is no need to periodically refresh the data in memory in this prototype. Because conventional logic analyzers and pattern generators have limited sample memory, it is necessary to frequently pause the execution of long-running tests to transfer data. If memory must be periodically refreshed, these pauses must occur between complete test cycles. This limitation is also present in conventional memory testers, so tests are commonly designed with the memory limits of the tester in mind. Removing the DRAM array eliminates the need altogether, allowing tests of any length or complexity to be executed.

Without a conventional memory tester, many useful signal characteristics could not be easily explored. To work around

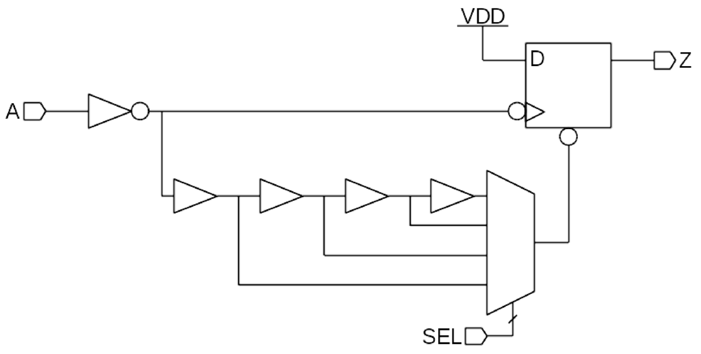

Fig. 2. Simplified adjustable pulse generator circuit

the limitations of the general-purpose instruments being used, circuits were designed to create signals internally. A network of scan chains and analog multiplexers allows the prototype to be dynamically reconfigured, adjusting signal timing, reference voltages and circuit behavior at areas of interest. Although inherently low-speed, the scan chains allow an enormous number of configuration registers to be modified during testing.

These registers perform a wide range of functions. Using adjustable delay and pulse generator circuits, they can shift signal timing internally with sub-nanosecond precision. Using analog multiplexers and transmission gates, they can select reference voltages from reference circuits, voltage dividers or external instruments. If supplied directly to logic, they can radically change circuit behavior. The adjustable circuitry is often very simple; figure 2 shows a simplified schematic of a pulse generator circuit used in the prototype.

Although these registers and adjustable circuits occupy a limited amount of space on the die, they are enormously powerful for research. The additional flexibility they provide results in a more robust prototype capable of adapting to the variability of the fabrication process and offering more insight into the design. Beyond simple adjustments, multiple implementations of a circuit can be selected during testing for evaluation. Highly configurable prototypes are valuable for research, since they allow more information to be gathered without additional fabrication costs.

A simple BIST was incorporated in the prototype, allowing addresses and command sequences to be programmed at low speed through one of the scan chains. The BIST can then be executed at the maximum design target frequency, by supplying an external high speed clock and asynchronous start/stop commands from the low speed scan chain inputs. Data outputs can be observed in real time with the appropriate test equipment or scanned out at low speed between high speed BIST runs. This allows a complete at-speed test characterization of internal memory functions with only one external high speed clock generator. If necessary, a built-in phase locked loop (PLL) could be utilized to operate beyond the capabilities of the clock generator.

B. Verilog Model

The hardware testing process begins with a model of the prototype written in Verilog. This model was automatically generated from the prototype's schematics, but several areas which could not be accurately netlisted required Verilog behavioral models coded to represent their functionality. In particular, analog blocks controlling signal timing and voltage levels were replaced with fixed delays and constants. In addition, some frequently-instantiated blocks were replaced with simpler models to reduce simulation time.

Because the model is generated from schematics of the prototype, its layout and signals are nearly identical to the hardware itself. The internal signals of the model closely match those of hardware, providing valuable visibility into the prototype which would be inaccessible in a behavioral model or the hardware itself. This visibility simplifies designing and debugging tests and ensures that tests are performing the expected operations.

The cost of this visibility is slower simulation times. Behavioral models offer significantly faster simulation times at the cost of limited visibility within the model. Combining both approaches provides the benefits of both: a behavioral model provides the canonical behavior of the device, while a netlisted model is used to explore deviations from the expected behavior.

C. Test Design

The representative Verilog design model is wrapped in a behavioral testbench which generates complex sequences of inputs during simulation. The testbench uses only the pins present on the chip itself, so the input sequences can be replayed on the pins of the physical chip to reproduce the simulation in hardware.

To simplify working with the model, a library of tasks and predefined registers was developed. Common operations like reading and writing from memory are performed through tasks taking address and data parameters. To simplify configuration during tests, the embedded configuration registers are visible as Verilog registers within the library. Changes to these registers are collected and programmed automatically before subsequent command tasks run. This simplifies the testbench code significantly, allowing a few lines of Verilog to produce thousands of cycles of inputs.

The testbench library generates output in a simple format suitable for processing by other programs. Each clock cycle, the library captures the state of all the model's inputs and outputs. Inputs and outputs that do not have meaningful values are marked as such. As test modes are changed, these changes are reported as well.

By capturing the outputs from the testbench, external software is able to reproduce tests in hardware. This software processes the input signals into a vector format that can be loaded into the pattern generator and makes note of the outputs and active configuration of the model. For functional testing, after the pattern generator runs, outputs from the hardware are captured from the logic analyzer. The outputs are compared, and results are stored along with the active test modes for later analysis. Other measurements may be taken instead, allowing power consumption or signal levels and timings to be captured.

D. Equipment, Clients and Servers

By necessity, the programs responsible for running tests are spread across a network, communicating by using a client-server model. Communication with instruments occurs across a wide range of interfaces with different requirements. In particular, the large amounts of data transferred by the pattern generator and logic analyzer necessitate high-speed connections. Ethernet provides ample throughput for these devices, while other interfaces would not.

Embracing a networked model makes testing much more convenient. It allows tests to be designed and written at an engineer's workstation, simulated on a high-performance compute server, streamed to the test equipment in the lab area for execution, recorded on a database server and finally analyzed from software running on the engineer's workstation.

Two server programs manage all the equipment required to run each test. The power supplies, source meters and clock generator are managed over General Purpose Interface Bus (GPIB) by a server running on a workstation in the lab area. This server accepts low-level commands that are passed to the test equipment. These commands include simple operations like changing power supply voltages, reference voltages and clock frequencies, as well more complex commands like measuring power.

A second server runs on a Tektronix mainframe with a logic analyzer, pattern generator and oscilloscope installed. Rather than accepting individual commands, it provides fixed execution pipelines which stream stimulus data in and response data out. Several additional operating modes are available, including looped patterns for measuring power consumption during complex operations.

A client program is responsible for setting up the entire test process. It launches the simulation, processes the simulation's output to configure test equipment and synchronizes configuration changes between the two servers. As it receives response data, it finds the associated metadata, then stores the stimulus, expected and actual response, and metadata describing the test in an external database for later retrieval.

Storing results in a database is convenient. The results are available to a variety of tools and stored in a consistent format, making automated analysis possible. Since each test is run against both a model and hardware, the results from both are stored side-by-side for comparison. Since libraries for communicating with database servers are available for most scripting languages, a wide range of software can be used for analysis.

III. Design Considerations

Designing a test environment requires forethought. First and foremost, the testing environment must be able to thoroughly measure all parameters of interest on the device. These requirements will determine which instruments and internal test modes are required and will affect the design of the environment as a whole.

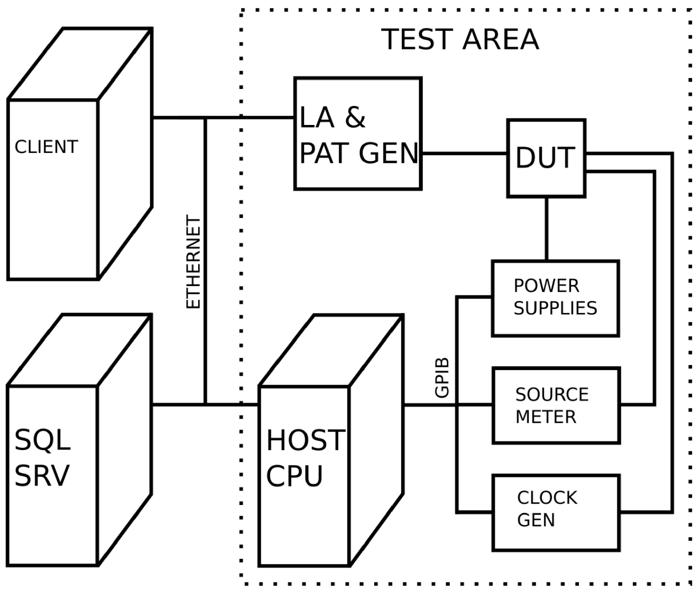

Fig. 3. Block diagram of test setup including test equipment, SQL server and client connection

A. Instruments

The characteristics of the instruments in use have a large effect on the capability of the test system as a whole. Instruments' maximum operating frequencies may limit high-frequency testing, which may be partially overcome using integrated circuit elements. Pattern generators and logic analyzers have limited memory and can transfer data at a limited rate. The maximum rate at which tests can execute is largely defined by these characteristics. Although enhanced triggering in the logic analyzer and more complex programs for the pattern generator can be used to overcome this limitation, these features may not be available to external software.

B. Hardware

In production, a device needs a limited amount of adjustability to compensate for variations in the production process. During the research and design verification phases, it is helpful to have a much wider range of adjustments available for experimentation. Designing a range of simple, adjustable circuits allows this to be realized.

A variety of adjustable circuits can be constructed. If signal timing is to be handled internally, delay circuits and pulse generation circuits are valuable. For amplifiers and other analog elements, adjustable voltage references are helpful. In addition, exposing reference voltages on one or more pins using analog multiplexers allows the reference to be precisely varied by external instruments rather than requiring large amounts of circuitry.

Using adjustable circuits where adjustment is not needed increases the size and complexity of the design and complicates simulation and testing by adding additional variables. Therefore, their use should be carefully considered. Engineer expertise and extensive simulation are necessary in these considerations.

A critical component of any design is documentation. It is especially important where adjustments are made by loading registers on scan chains. Each scan chain can contain a large number of registers, so the position of each register in the chain may be difficult to establish. Keeping documentation of scan chain connectivity is necessary for testing to be performed.

In addition, it is helpful to document simulation results to provide a starting point for hardware testing. While having a wide range of adjustments available allows more thorough testing, each available adjustment is another variable that must be considered when configuring the hardware. Simulation data is invaluable in determining the physical meaning of each adjustment setting and in establishing a reasonable configuration during testing.

C. Software

Conventional memory testers are designed to execute tests very quickly to minimize test time during production. General-purpose instruments may execute tests much more slowly than dedicated test equipment, so the software controlling them needs to be designed with the limitations of the instruments in mind and optimized to allow rapid test execution. A significant amount of vector data must be generated and transferred to execute tests. Speeding up the handling of this data is the most significant target for optimization.

To execute a test in hardware, a simulation must be run, its output must be processed into stimulus and finally the instruments must replay the stimulus. If these steps are performed serially, it will take a long time to set up each test. Pipelining the process hides much of this start-up time. If stimulus is generated on-the-fly as the simulation runs, hardware testing can begin almost immediately. The disadvantage of pipelining is that the Verilog simulator may become a bottleneck if the instruments outpace it, keeping the instruments busy longer than necessary and preventing other tests from being executed.

Transmitting large chunks of vector data across the network may take a significant amount of time. While many modern instruments are often equipped with Ethernet, USB, or other high-speed ports, these ports are not ubiquitous. Older instruments may have very slow ports, or may require slow external adapters. On these devices, data transmission may cause significant delays. These delays can be masked by transmitting vectors before they are needed, at the expense of more complex communication protocols.

Between each chunk of vector data, the logic analyzer's sample memory must be captured and new vectors must be loaded into the pattern generator's memory. Both operations take a significant amount of time. Arranging for the two operations to take place simultaneously significantly reduces the time required between chunks, providing a large benefit to overall execution speed. However, the software required to manage this parallel execution is necessarily complex.

In a networked test system, the protocols used for communication are important. Although a simple, command-oriented protocol is simple to implement to control a single instrument over the network, it quickly becomes burdensome as multiple instruments are added. In such a protocol, a controlling program issues simple commands to servers responsible for managing instruments and waits for the completion of a command. Such a protocol is natural to implement, since software libraries for controlling instruments provide low-level commands for each function of the instruments. Exposing these commands over the network is simple, but is necessarily slow. Executing multiple commands in parallel requires additional synchronization and this synchronization quickly becomes unmanageably complex.

A protocol which gives servers more autonomy makes parallel execution more manageable, but is complex to implement. In such a protocol, individual commands are not sent. Instead, one or more fixed execution pipelines are used. For example, a pipeline might consist of receiving vector data for a pattern generator, running the pattern generator and collecting outputs using a logic analyzer and responding with the logic analyzer's recorded outputs. The server manages this pipeline itself, simplifying the client's view of how tests are executed. However, this makes it difficult to synchronize other instruments or add new execution modes.

A more complex protocol which treats the execution of a test as a series of events offers advantages over other protocols. The client does not need any knowledge of the commands that will be executed. Instead, it translates the execution of the test into a series of events with associated sequencing information and other related data. The servers controlling each instrument can then track the progression of events and execute the necessary commands at the appropriate moment. This eliminates the need for explicit synchronization in the client and allows all instruments to be managed in parallel. Although the protocol is complex to implement, the remainder of the software is dramatically simpler, inherently parallel and easy to extend with additional instruments and modes of execution.

General-purpose instruments allow a variety of data to be captured and this data is useful for research. The format used to store results in the database needs to be flexible in order to take full advantage of the instruments. It also needs to be easily extensible to allow additional instruments or test designs to be added. Such a format will necessarily be more complex than a spreadsheet-like layout with columns for each parameter. Although they require more effort to work with, relational databases are a natural choice for storing the wide range of data acquired through testing.

Whether simple file formats or complex database schema are used, analysis of test results will need access to the stored results. File-based spreadsheet formats like CSV and database formats like SQLite are relatively easy to work with, but managing the files can be cumbersome. Full-featured database servers like MySQL simplify managing the result data while keeping it simple to access. Software libraries are widely available to communicate with database servers from within applications like R, allowing powerful analysis to be performed easily.

IV. Disadvantages

A. Low-Volume

Some elements of this approach are focused on low-volume prototype testing, such as the utilization of low-resource/low-speed logic analyzers for pattern generation. This is not meant to replace the high-performance instruments used in production for maximum throughput. However, many elements of this solution such as on-chip high-accuracy timing and voltage variations are reusable for product manufacturing and can reduce the amount of resources needed per chip, for higher parallelism in testing.

B. Software Development

Operating numerous instruments manually to perform tests is impractical. For an arrangement like this to work, automation is necessary. This means that before testing can begin, a significant amount of software must be written to tie the instruments and simulation software together into a cohesive test system. This software quickly becomes complex in order to handle communication between instruments across a network, which is not typically part of standard test development.

C. Simulation Dependence

Using simulation software to generate vectors for testing has several drawbacks. Although Verilog simulations can execute far faster than SPICE simulations, using simulations to generate expected outputs limits testing throughput. Testing software can reduce the effects of this by beginning hardware testing while the simulation is still running, but simulation speed is a dominant factor in how quickly tests can execute.

Alternatively, if the runtime for stimulus generation is of concern, a simpler model can be used to create stimulus and expect data. While this model is similar to a memory tester's approach of coding expect data and relying on engineers' understanding of the specification, it will allow a high speed test code compilation.

A more significant problem with simulation is that the model itself must be correct. Although automated schematic netlisting tools may accurately reproduce a schematic in Verilog, they may not accurately capture its behavior. This is particularly challenging when analog behavior is intended. Timing information may not be available to the Verilog simulator, preventing it from correctly modeling gate delays. Verilog's rudimentary drive strength model may cause some circuits to behave incorrectly. While Verilog-A allows for analog circuits to be modeled, designing these models may be more time-consuming than producing a behavioral model. When using netlisted models, care must be taken to ensure that the Verilog model of the design behaves correctly.

V. Advantages

A. Testing Against the Model

Typically, a behavioral model of the design is created early in the design process. This model defines the correct behavior which the simulated design and hardware must conform to. By using it to generate expected data, it is possible to test hardware directly against the model, with no opportunity for misinterpretation or error.

B. Cost

Using general-purpose test equipment that is readily available on the secondary market significantly reduces testing costs. For prototypes designed for low-frequency operation, less expensive, less capable equipment can be used. Because general-purpose equipment is inherently reusable, it will remain useful after the conclusion of testing.

Conversely, conventional memory test equipment loses its value rapidly. As newer, faster testers become available, the older, slower testers quickly depreciate in value. If state-of-the-art memory is to be designed, testers from past generations may be too slow to be of use.

C. Flexibility

Using independent test equipment as part of a platform interconnected with software is very flexible. As additional equipment is needed, it can be plugged into the network and used. Since the protocols used for communication can be modified as needed, they can be expanded to accommodate the new equipment.

D. Convenience

For research, being able to easily write tests and quickly execute them is invaluable. Tests are written entirely in Verilog, which is a comfortable language to use when working with digital logic. A wide range of tools can work with Verilog to make debugging tests and test failures much simpler. This test setup takes advantage of these benefits to make the hardware very easy to work with.

VI. Conclusion

The approach outlined avoids the drawbacks of traditional memory testing. By avoiding the need for comprehensive BIST circuitry, development times are reduced and by replacing conventional memory testers with low-cost, general-purpose instruments, testing costs are significantly reduced. Using simple circuit additions and readily available test equipment, a test system can be constructed to meet the unique demands of novel memory design research. For small companies on a very limited budget, practical approaches to testing open the doors to innovation.

References

[1] Article, Rick Nelson, 09/01/04 https://www.edn.com/design/test-and-measurement/4378511/Tabletop-testers-From-prototype-to-production-

[2] International Technology Roadmap for Semiconductors (ITRS) 2009 http://www.itrs.net/Links/2009ITRS/Home2009.htm

[3] Article, Jeff Dorsch, 10/18/17 https://semiengineering.com/challenges-and-opportunities-in-memory-test

[4] Georgi Nedeltchev Gaydadjiev, "Testing of Modern Semiconductor Memory Structures", Sep. 25, 2007 Delft.