

METHODS FOR MEMORY TESTING

by

Jing Plaisted

A project

submitted in partial fulfillment

of the requirements for the degree of

Master of Science in Engineering, Electrical Engineering

Boise State University

October, 2003

The project presented by Jing Plaisted entitled Methods for Memory


Testing is hereby approved:



Advisor 11/11/03
Date



Committee Member 11/11/03
Date



Committee Member 11/12/03
Date



Graduate Dean 11-12-03
Date

ACKNOWLEDGEMENT

I would like to sincerely thank Dr. Baker for his guidance, patience, and all the support throughout this project. I also like to thank Dr. Hartman, Dr. Smith for their support and help. My special thanks go to Albertson Library staff for providing the materials that I need in timely manner. Last but not least, I like to thank Kent, Ben, and Johnny for their help and time.

TABLE OF CONTENTS

APPROVAL PAGE.....	ii
ACKNOWLEDGMENTS.....	iii
INTRODUCTION	1
Why Do Memories Need To Be Tested?.....	1
Current Test Status.....	1
TEST DEFINITIONS.....	4
Quality Measure Standard.....	4
Defects	4
Faults.....	4
Fault Coverage.....	5
Failures.....	5
Fault Models	6
Stuck-at Fault Model.....	6
Delay Fault Models.....	8
TESTING TYPES.....	9
Functional Test.....	9
Structural Test.....	9
Combinational Test.....	9
Sequential Test.....	10
TEST EQUIPMENT AND TEST PROGRAM.....	11

Automated Test Equipment (ATE).....	12
Automated Test Pattern Generation (ATPG).....	13
Pseudorandom Test Generation.....	13
Deterministic Test Generation.....	14
BOUNDARY SCAN TEST.....	15
Boundary Scan Test Motivations.....	15
Boundary Scan Components and Their Functions.....	16
Boundary Scan Cell.....	18
Boundary Scan Architecture.....	19
Boundary Scan Applications.....	20
Testing Tools.....	21
I _{DDQ} TEST METHODOLOGY.....	23
I _{DDQ} Test.....	23
The Need of I _{DDQ} Testing and its Shortcomings.....	24
I _{DDQ} Test Models and Methods.....	26
I _{DDQ} Test Methods.....	26
I _{DDQ} Design Concerns.....	27
I _{DDQ} Challenges and Future Trends.....	27
BUILT-IN-SELF-TEST (BIST).....	29
BIST Advantages and Disadvantages.....	29
Testing Types and Algorithms.....	31
Testing Types.....	31
Testing Algorithms.....	31

BIST Applications	32
BIST Design Concerns	33
SUMMARY.....	35
REFERENCES	36

INTRODUCTION

Why Do Memories Need To Be Tested?

Many scientists and engineers are striving to decrease the die size and lower the development cost so that the semiconductor industries are able to increase their productivity and profits. With Integrated Circuits (IC)'s size shrinking smaller and smaller and improved layout topologies, more condensed memory arrays are assigned to much smaller chip area, it becomes more challenging to test memory devices, such as, flash, DRAMs, SRAMs, embedded memories, and other critical memories for high defect coverage and stuck-at faults. In order to maintain excellent product quality, to achieve high standard reliability, and to meet customers' satisfaction, many advanced test methods have been developed or are under development. The basis for these efforts are to meet the requirements of consuming less time to perform tests at different levels and have higher defect or fault coverage. This paper will provide an overview of some current memory test techniques.

Current Test Status

Design-for-test (DFT) is strongly emphasized in the semiconductor industry because it can contribute to a major portion of reducing the overall product cost. The goals of reducing test cost include: decreasing test time, simpler test steps, and less testing data (a.k.a. data vectors) need be applied and generated. Therefore, *testability* is one of the major concerns when a device is under design. The contemporary testing is

mainly focusing on functional testing in order to detect faults that are caused during the processing. In some cases, the scan techniques and Automatic Test Patterns Generation (ATPG) are used together to achieve higher fault coverage. Currently, the memory testing techniques that are used in the industry are I_{DDQ} test, which is current based measurement test, boundary scan test, Built-in-Self-test (BIST), and other voltage based measurement tests. The tests may be performed on different hierarchy levels, which depend on a customer's specification. If one test method has lower defect coverage, a combination of different test techniques may be applied on the device under test (DUT) to achieve a higher percentage of detecting defects and faults. Many materials and concepts that are present in this paper are learned from [Crouch] and [Klenke98] unless they are specified otherwise. The remaining sections of this paper are organized in the following order:

- Chapter two will cover some of the test definitions, including defects, faults, fault coverage, fault models, and failures.
- Chapter three introduces different types of testing, functional test, structural test, combinational test, and sequential test along with their models.
- Chapter four will cover testing equipment and testing programs including Automatic Test Equipment (ATE) and ATPG.
- Chapter five will discuss the boundary scan test technique and its applications in a detailed level.
- Chapter six will introduce the quiescent current (I_{DDQ}) based testing technique and compare it to other testing techniques.
- Chapter seven will cover BIST, including its advantages and disadvantages, applications and some of the design concerns.

- Chapter eight will summarize the testing techniques that are discussed in this paper, the testing problems, design suggestions, and the future trend for testing.

TEST DEFINITIONS

The basic rationale of testing is to check whether the device under test (DUT) including chips, printed circuit boards (PCB), and system can produce expected results when known inputs or “input stimuli” are applied. In general, a simple test program will produce results like “pass” or “fail,” while a more advanced testing procedure will not only verify DUTs pass or fail, but also determine the type of defects or faults, and report their locations. This information is important when repair is needed. If the defects or faults happen in a pattern, then it indicates an improvement in the process is necessary [Chen2001].

Quality Measure Standard

Defects

The physical flaws that occur in the chip are called defects. In a CMOS process, shorts between two terminals, gate oxide shorts or pinholes, and metal trace shorts or opens are considered defects [Klenke98] and [Sidorowicz2002]. These defects can usually be identified with functional testing, which will be discussed in next chapter.

Faults

A fault is a logic error that is caused by a defect and can be modeled by a fault model. Many defects may exist in a circuit, but only some of these defects result in a fault. There are many possibilities that can trigger faults in a circuit, Fig. 1 illustrates potential areas that can cause faults.

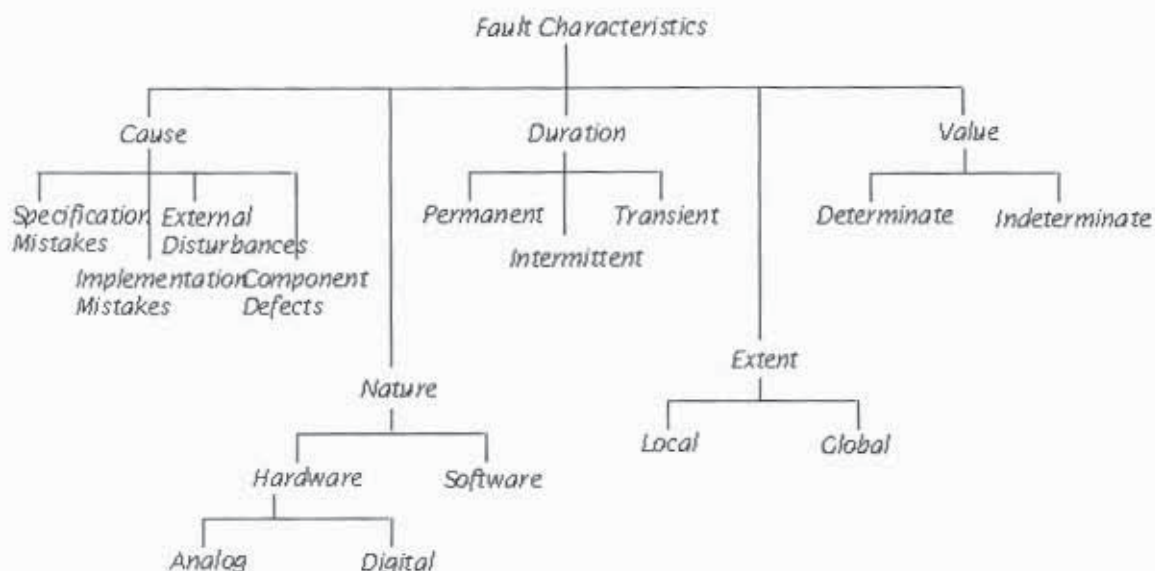


Figure 1 Areas that can cause faults (Ref: [Klenke98])

Fault Coverage

Fault coverage is defined as the ratio of the detected faults to the total fault amount. One must be careful when determining the fault coverage because the total fault can be different depending on the circuit design and testing tools. Some designs may only cover a certain number of faults, in this case, the relative fault coverage (using total detectable fault as total fault) is higher than the absolute fault coverage (using all possible faults as total fault).

Failures

It may not always cause a failure when a defect is found in a chip by a fault model. A failure should obtain the characteristics, such that the fault model can evaluate the fault and provide a judging standard for determining a failure.

Fault Models

The frequently used fault models include: stuck-at faults, stuck-open faults, bridging faults, delay faults (transition fault), and pattern-sensitivity faults in dynamic memories [Klenke98], [Sidorowicz2002]. The examples often seen in a circuit are stuck-at-0 (a signal tied to ground) and stuck-at-1 (a signal tied to VDD or power).

Stuck-at Fault Model

There are many types of stuck-at fault models. We will focus on the single stuck-at fault model and multiple stuck-at fault model.

Single Stuck-at Fault model

There are some assumptions for using this model:

- Only one fault exists at a time.
- The fault is stable.
- The fault is either stuck-at-0 or stuck-at-1.
- The gate function should be predictable when the fault exists.

In order to detect a single stuck-at fault, certain conditions need to be satisfied before using the general steps of the model. The conditions to be satisfied include the testing node be observed, and a “golden board” is available to compare the tested results to the “ideal” or expected results. Some of the manufacturing industries have been using “golden board” as part of their normal procedure at the testing stage. The node to be tested is made active by putting restrictions on all the other nodes’ value that can affect the output. The input stimuli – all possible combinations of logic “0”s and “1”s are

applied to the DUT. A fault is detected when the output and expected values or logic are different.

A simple example would be an inverter in a digital circuit. If input X is stuck-at logic 0, then the output Y will always produce the same output, logic 1, even when a 1 is applied to X. The correct result appears when $X = 0$ but incorrect when $X = 1$. If the input is stuck-at logic 1, then the output Y will always produce the same logic 0 even when 0 is applied to X. The output is correct when $X = 0$ and incorrect when $X = 1$. Lastly, if the output Y is stuck-at logic 0 or 1, then the output of the inverter will be logic 0 or 1, respectively, no matter what the input is.

There are many advantages of using the single stuck-at fault model: The model is easy to use with well developed algorithms and fault simulation; higher defect coverage – about 90% of defects in a CMOS process can be achieved; stuck-open or shorts can be “mapped” to single stuck-at fault model. Because of these reasons, the single stuck-at fault model is well accepted in testing digital circuitry.

Although this model can detect a high percentage of defects, the drawback of this model is that it is not able to cover defects one hundred percent of the time if the logic has multiple stuck-at faults or if the fault is transient. Therefore, multiple stuck-at fault models have been developed to overcome part of the problem.

Multiple Stuck-at Fault Model

The assumptions to use multiple stuck-at fault model are similar to that of single stuck-at fault model, except that the faults are multiple instead of only one in the DUT. Although this model has higher defect coverage when it is used together with the single stuck-at fault model, there are many drawbacks of this model. The model is not well

developed due to the fact that the defect coverage does not increase dramatically and it is a much more complicated testing technique.

Delay Fault Models

The conditions to use a delay fault models are that the logic function of a testing device is correct and the delay caused by inconsistent processing exceeds the specified value. The transition delay and path delay fault models are commonly used for the delay fault model.

The Transition Delay Fault Model

The transition delay model (a.k.a. gate delay fault model), is a timing fault model. It is used to detect a single gate delay by applying the slow-to-rise or slow-to-fall signal to the input. The transition delay fault is detected when the time from input to output surpasses the defined value. If the delay is long enough, it can be modeled as a justified version of the single-stuck-at fault model.

The Path Delay Fault Model

The path delay fault model is used to determine the total delay of a definite circuit path in combinational circuitry. The analyzed objective for this model is the path through multiple gates rather than a single gate when compared to the transient delay model. The benefit of using this model is that it can detect multiple gate delay faults along the path, as the model's name indicates. Then again, the trade off is that the test can be more complicated since there are many paths from point A to point B in a complex circuit.

TESTING TYPES

Functional Test

Conducting a functional test is to check if a device's designed function meets the desired specification. To achieve this goal, a set of known inputs are applied to evaluate the output at the gate level or behavioral level of the design. The standard requirement for functional test is 100% accuracy.

Structural Test

The purpose of the structural test is to check the chips' topology correctness at gate-level. The stuck-at fault model is used to conduct this kind of test. An example would be forcing an opposite value on a stuck-at fault suspicious node (apply a 0 when it is stuck-at-1 fault), and then switching the inputs to such a value that the circuit would produce the correct output. This test can detect the fault point when the circuit fails to yield the correct output if the appropriate inputs are applied. High fault coverage (95% - 99.9%) is expected in the semiconductor industry.

Combinational Test

The Combinational Test is used to check if the circuit can produce the correct result when all possible input combinations (if the device has N input, then the total possible input combination is 2^N) are applied to the DUT. This test is normally performed in order to characterize a logic circuit for "timing or logic operation."

Sequential Test

The Sequential Test is the combination testing of sequential elements and combinational circuit test. This test is one of the most difficult to conduct, as well as, time consuming because the sequential states are embedded within the circuit. Even when applying all the exhaustive testing inputs to the circuit it will not be successful enough to generate a full coverage of the output. A circuit with an M element state machine and N inputs will require as many as $2^{(M+N)}$ tests. A better solution to this complicated test is to use the scan technique or ATPG algorithm, which will be covered in more depth in the second section of the next chapter.

TEST EQUIPMENT AND TEST PROGRAM

The test equipment (or tester) and test programs vary depending upon the variety of the memory and quality requirements. The trade off for the sophisticated testers, normally performed by trained engineers, is between degree of operational difficulty, and high fault coverage. On the other hand, memory diagnostic software is inexpensive; therefore it offers a lower cost testing tool option. Its purpose is to generate test patterns to detect potential problems during memory testing. The major problem with using the software is the possibility that it will not function properly or may even become ineffective when a fatal memory fault or failure exists in the DUT (device under test). In short, the goal is to use a simpler and faster tester with demonstrated cost-effectiveness, and most importantly, to catch the maximum amount of memory faults and defects.

Automated Test Equipment (ATE)

As previously mentioned, both hardware and software can be used to test memory chips, boards, and systems. In general, Automated Test Equipments (ATE) are used to test memory chips during their final process. However, the hardware memory tester is more difficult to build into the tester because of its complexity and various test characteristics. This is the main reason why the high-end ATE can cost a million dollars or more. Micron is one of the companies in of the semiconductor industry, which utilizes the ATE for testing memories.

The three components of the Integrated Circuit ATE are shown in Fig. 2: numerous “channels with memory depth,” multiple clock generators, and multiple power supplies.

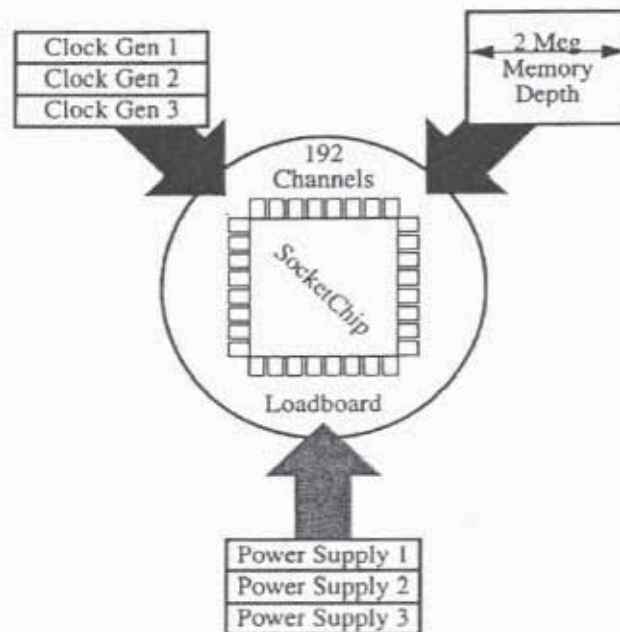


Figure 2 ATE components (Ref: [Crouch])

The cost for ATE is dependent on the size of the memory being tested, process usage, and the precision of the results. The expense for ATE increases when testing the larger size memory due to considerable amount of vectors that need to be generated and the circuitry is more complicated. As we expect, lower usage of the ATE will increase overall testing cost per unit. There are many ways to reduce the ATE cost, such as, making the testing program simpler, using smaller vectors which can also reduce the testing time.

Automatic Test Pattern Generation (ATPG)

If the memory circuit is considerably large and complicated, the use of Automatic test pattern generation (ATPG), to generate the test vectors, is essential in order to achieve high fault coverage since it is not practical to manually apply all the test vectors.

The main techniques used in the ATPG to detect the stuck-at faults are pseudorandom and deterministic test generation. In addition, algorithms that are used include: ad hoc, D algorithm, PODEM, and others.

Pseudorandom Test Generation

When the probability of identifying any detectable faults is the same, then it is called a pseudorandom test generator. The pseudorandom test generation is usually conducted at the beginning of the test and is used to detect “easy-to-detect” faults. The pseudorandom test is efficient because it has fairly good fault coverage (nearly one-hundred percent) using smaller test vectors. A Linear Feedback Shift Register (LFSR) is used to generate the pseudorandom patterns.

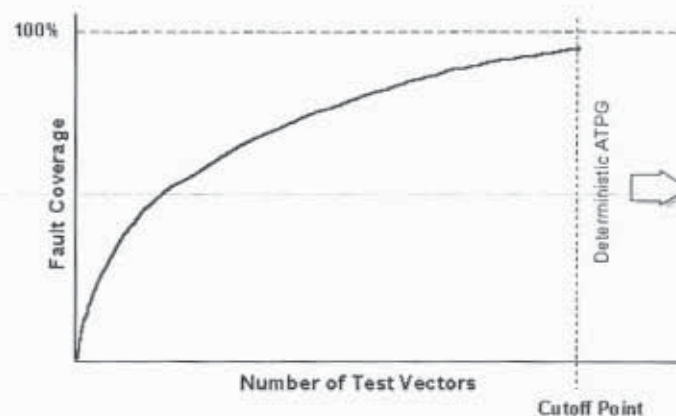


Figure 3 Pseudorandom fault coverage (Ref: [Klenke98])

Fig. 3 shows that fault coverage will no longer increase significantly when the quantity of test vectors reaches a certain level (pseudorandom testing can not detect any additional expected faults while increasing the number of test vectors). This is where the deterministic ATPG is needed to detect the rest of the expected faults.

Deterministic Test Generation

The deterministic test is defined as the repeatable response of the unit-under-test. The deterministic test generation is used in order to achieve higher fault coverage since the pseudorandom test can not cover all possible faults. As a result, the manufacturers compromise chip area (more overhead) and the related excessive cost for the higher product quality and reliability [BIST of Memory].

BOUNDARY SCAN TEST

A successful test design basically has three criteria: high quality, quick results, and desirable economics. To achieve these goals, the Joint Test Action Group (JTAG) was formed in late 80's. The group of professionals, who came from circuits and electronics product manufacturing and test engineering fields, defined four wire test bus that are required for boundary scan architecture [Gopel electronic] and an optional common asynchronous reset signal [1149.1-2001]. The result of their work became the industry standard, also known as IEEE Std 1149.1.

Boundary Scan Test Motivations

Boundary scan architecture became IEEE Std 1149.1 in 1990 in order for the test bus to be utilized effectively as a universal test platform [Gopel electronic]. With the increased usage of Ball-Grid-Array (BGA) packaging and other newly developing packaging styles, it became more challenging to access each component for testing, therefore, boundary-scan became well accepted for board manufacturing testing [Wondolowski99]. Since IEEE 1149.1 was published, a few extensions have been developed and released, namely, IEEE 1149.1-b, IEEE Std 1149.4, used for analog testing, and IEEE Std 1149.5, used for testing at system level [Gopel electronic].

Boundary scan made it possible to use the same test for different boards or at the different levels if the device is scan capable [Wondolowski99]. It makes the higher level testing more efficient, yet less costly, since it is now possible to skip testing the "bug-free" boundary scan portion instead of building a much more complicated test fixture to

cover every segment. The ability to access TDI (boundary scan test data in) and TDO (boundary scan test data out) is highly valuable because the functional tests and debugging can be conducted in an easier way now. Another benefit of using boundary scan is that it made the difficult testing tasks easier to handle in a software form [Boundary Scan]. The Boundary scan test can still be more efficient, even when a board contains some non-boundary scan logic, since the software is able to facilitate the testers to access them via boundary scan cell.

However, like other testing techniques, boundary scan is not perfect since the area, speed, and testing overheads are necessary for the test design and these elements can slow down the speed and increase the chip area.

Boundary Scan Components and Their Functions

Originally, boundary scan was designed to test the chip and extend its test availability to the next integrated circuit level so the ATE can be used repeatedly at different levels. The Boundary scan technique provides overall “control and access to the boundary pins,” eliminating the requirement of the bed-of-nails or other testing equipment [Gopel electronic], as shown in Fig. 4.

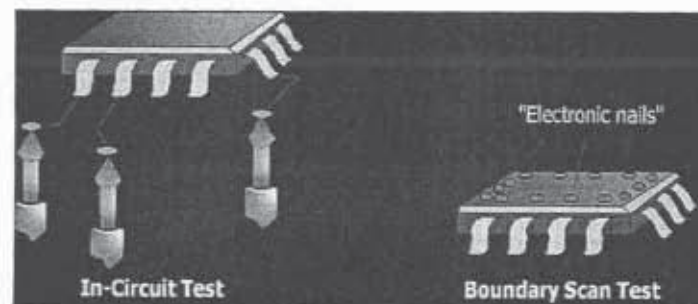


Figure 4 Boundary scan transforms the physical pins to “electronic nails” (Ref: [Gopel electronic])

The basic boundary-scan structure includes shift-register, which is positioned in a scan cell and next to the physical pins so that the signals can be stored and tested at the boundary ([Ref: 1149.1-2001]).

The boundary-scan test logic consists of three parts: the test access port (TAP) controller, the instruction register, and the data register. The TAP controller, a 16-state finite state machine (FSM), see Fig.5.2, is used to generate the clock and to select the correct signals. The state machine changes states only at the rising or falling clock edge. The control signal for state transition is the test mode signal (TMS). The instruction register functions as the decision matrix, and gives direction on corresponding steps based on stored instructions. Series of input stimulus are stored in the data registers then executed based on the operation commands.

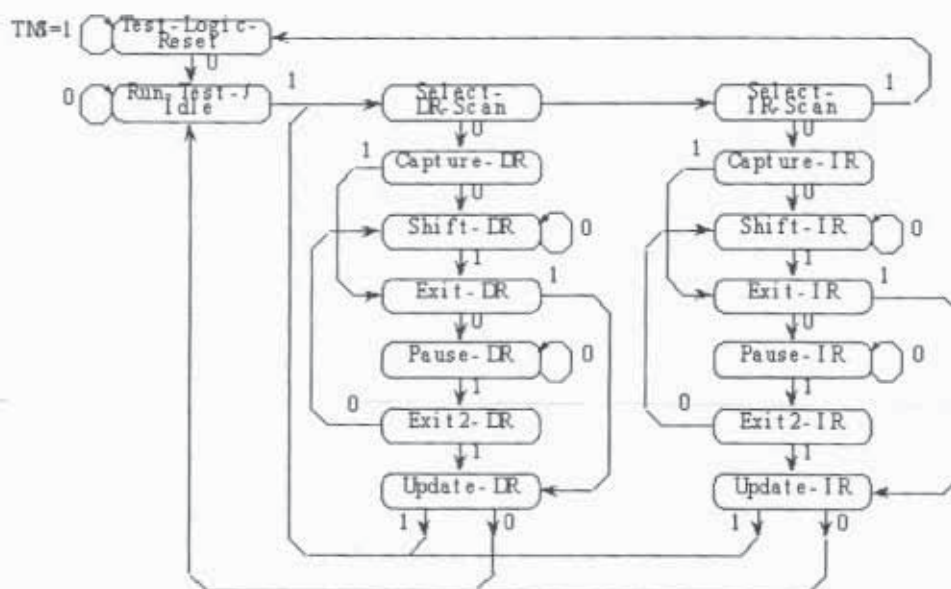


Figure 5 TAP controller state machine diagram (Ref: [1149.1-2001])

Boundary Scan Cell

Fig. 6 shows an example of a boundary scan cell. Every boundary scan device consists of one or more boundary scan cells. The most common modes used in a cell are: normal mode, scan mode, capture mode, and update mode. The control signal, MODE Control, selects the functional mode or test mode according to the operating instruction but not both at the same time. It is called normal mode when the functional mode is selected and the input signals simply pass through. The scan mode is when Shift DR selects SIN, SIN (Scan IN) goes to QA then to SOUT (Scan OUT). If Shift DR selects IN, then QA gets the value of IN, which complete the capture mode. Under the update mode, OUT gets the value of QA.

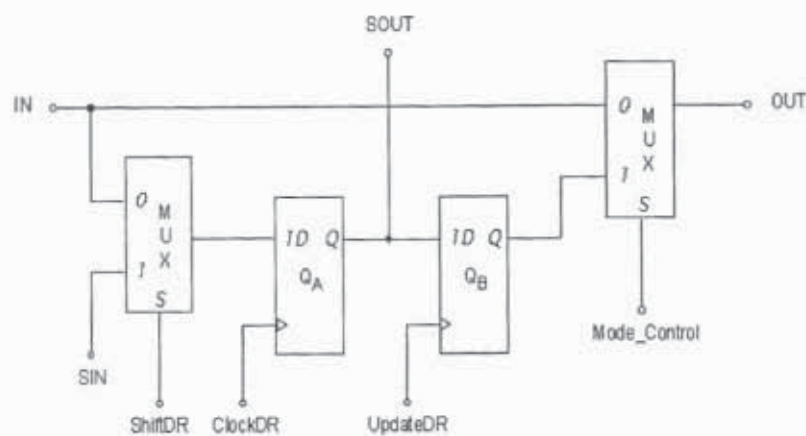


Figure 6 An example of a boundary scan cell (Ref: [Klenke98])

The connections between scan cells can be serial or parallel or a mixture of the two. It is considered a serial connection as shown in Fig.7, if TDO and TDI are tied together as a chain. The advantage of the serial connection is that less chip area is needed. For parallel connection, the benefit is that it can increase the circuit reliability if the

design is structured so the system can still work properly, even when a defective scan cell exists somewhere in a circuit. Another advantage of parallel connection is that it allows the debugging and fault analysis to be performed during normal operation [1149.1-2001]. The trade offs for having high reliability is that more chip area is needed, more complicated designs are required, and the increased burden on test design engineers [Boundary Scan].

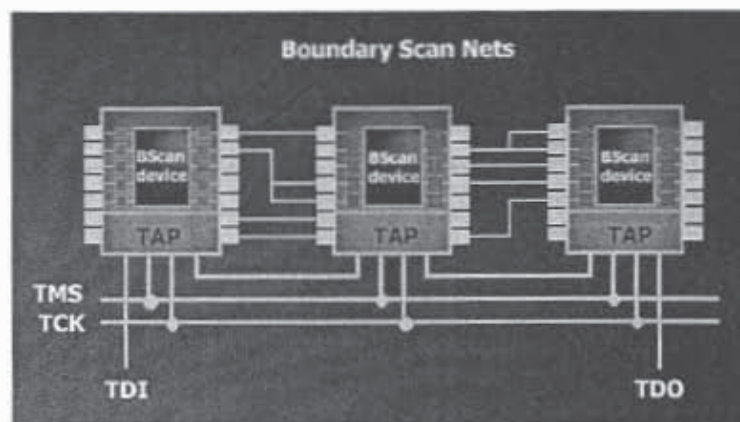


Figure 7 An example of boundary scan cell serial connection (Ref: [Gopel electronic]).

Boundary Scan Architecture

Fig.8 shows a chip containing boundary scan architecture and the connections of the four-wire test bus, TCK, TMS, TDI, and TDO. Other testing techniques, such as, BIST may be used as well in the same chip. In [1149.1-2001], the purposes of these test bus are as follow:

- TCK is the test clock; it is used in TAP controller and other registers
- TMS is the test mode select; it controls TAP controller's state machine.
- TDI is the test instruction and data input.

- TDO is the serial test data and instructions output; TDI and TDO are not active at the same time since TDI is updated at TCK's rising edge and TDO is updated at TCK's falling edge.
- TRST, a common asynchronous reset, is optional to extend the architecture [1149.1-2001].

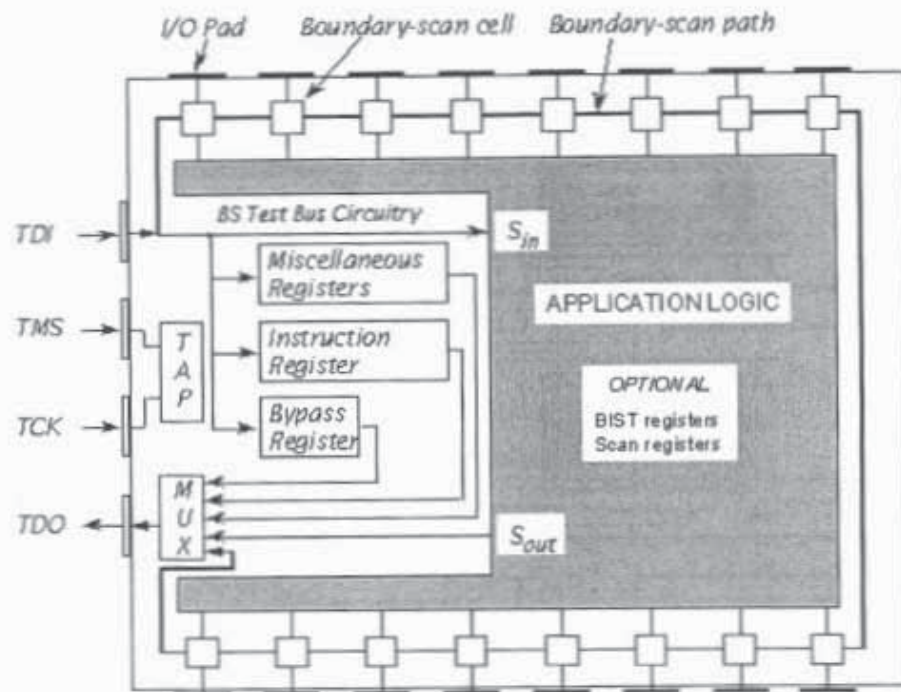


Figure 8 Architecture of a boundary scan capable chip (Ref: [Klenke98])

Boundary Scan Applications

Since 1149.1 became the standard, it made testing easier at these different levels: components (or chips), modules, PCBs, and systems. This is important because some modules and PCBs, in many cases, need to be integrated with other boards to make a system level product. In Wondolowski, Bennetts, and Lay's research, they explored the

method known as “daisy-chain” (daisy stands for data acquisition and interpretation system). “Daisy-chain” refers to a connection between the TDO of one boundary scan capable device to the TDI of another in a hierarchy structure. The first device and the last device are placed in an area that allows easy access to the higher level testing. For example, in Fig.9, the upper right corner is connected to the serial data input and the lower right corner logic is connected to the serial data output.

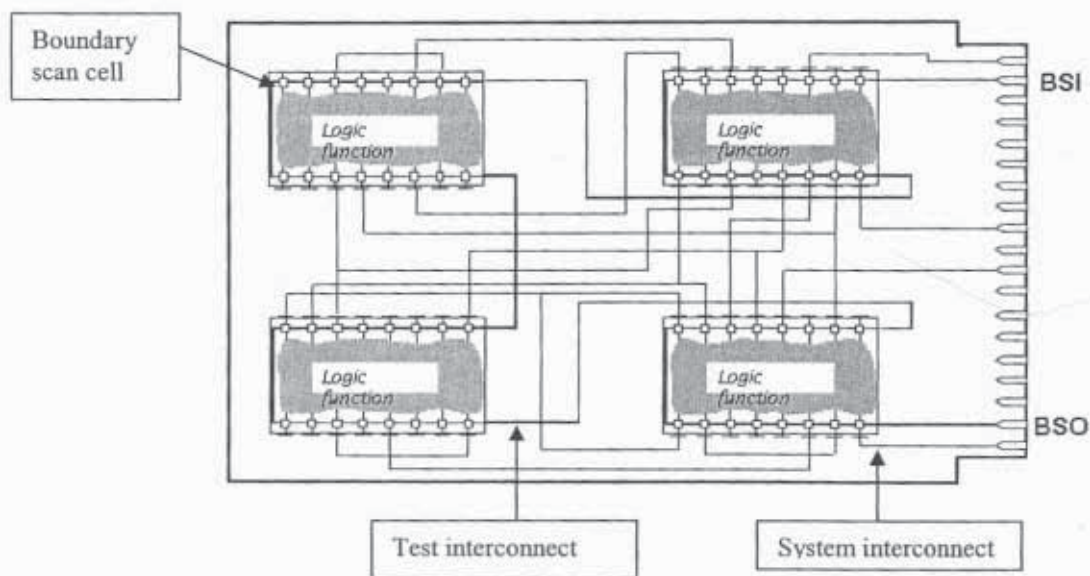


Figure 9 Boundary scan design in board level (Ref: [Klenke98] and [1149.1-2001])

Testing Tools

The optimum testing tools are those that are capable of use with a maximum number of products in order to reduce the overall manufacturing cost. However, this is not possible in every instance, especially when a new testing technology is introduced into a manufacturing environment and because some vendors cannot provide boundary scan compliant products as the system demands. In addition, if the test vectors can be

generated automatically, the tests are more user friendly. Therefore, some software is used to test the product combined with the use of boundary scan. This makes the test efficient by allowing access to the non-scan devices via the boundary scan cell.

I_{DDQ} TEST METHODOLOGY

I_{DDQ} testing technique is a methodology to detect faults and defects with quiescent currents at the transistor level in CMOS devices. In this chapter, I_{DDQ} testing and its perspectives will be discussed. The chapter will discuss some of the design issues and future testing development with this unique technique.

I_{DDQ} Test

The main difference between I_{DDQ} testing and voltage-measurement based circuit testing methodology is that the I_{DDQ} test is based on current measurement. The faulty devices' quiescent current increases several orders of magnitude compared to a defect-free device, therefore the defect is able to be detected. This technique assumes that the fault-free CMOS circuit consumes very little current (the range is within fA) in the static state. I_{DDQ} testing became one of the well developed and ubiquitous current based techniques.

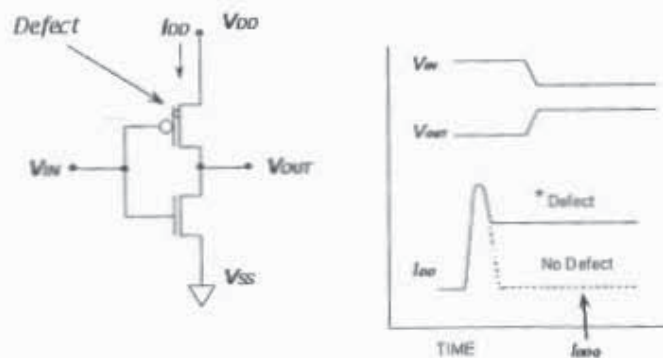


Figure 10 Quiescent current shows physical defect effect (Ref: [Klenke98])

An example of using I_{DDQ} testing to detect a gate-source short, in the PMOS of an inverter is shown in Fig.10. I_{DDQ} increases significantly compared to the defect-free device. When the input voltage changes from high to low, the PMOS turns on and pulls the output voltage to high. I_{DDQ} could increase several orders of magnitude because the short enables the current to flow from source to gate. We notice that whether this particular defect exists or not, the output voltage is high for both cases. However, the I_{DDQ} test can determine this defect while the voltage measurement method does not. It gives the impression this is not as critical at first glance since the defect does not affect the inverters function. In fact, the ability to detect this kind of defect is important because the defect could seriously degrade the product reliability, and consequently, cause a failure in the future.

The Need of I_{DDQ} Testing and Its Shortcomings

The I_{DDQ} test presents a number of benefits. First, I_{DDQ} can detect many defects in the CMOS circuit, that many other methods cannot, such as bridging faults, gate oxide defects, shorts, some delay faults, and some open faults [Rajsuman2000]. These defects can cause the current to increase dramatically (several orders of magnitude) compared to quiescent current. For example, I_{DDQ} can detect source-gate shorts as mentioned before, but the stuck-at fault would not detect this defect. However, this method cannot detect some of the open faults depending on the circuit design [Rajsuman2000].

Another benefit of I_{DDQ} is that this current-based measurement is effective because all of the nodes in a circuit are observable but not necessarily all controllable so that the fault is "activated." Therefore, the defect is evident and, as a result, the fault can be detected without the requirement of propagation.

The I_{DDQ} test advantages can be concluded as “ I_{DDQ} testing can be described as a low-cost, high quality, supplemental test” [Rajsuman2000].

Fig. 11 shows that using I_{DDQ} can achieve high fault coverage with a low number of vectors (less than 20 in the case of bridge faults).

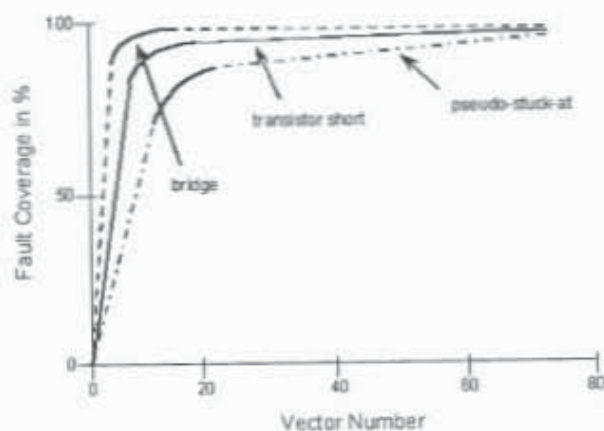


Figure 11 I_{DDQ} fault coverage vs. vector number (Ref: [Klenke98])

Like other testing techniques, I_{DDQ} has some negative sides as well. Since the test is based on a very small amount of current, during the test, the DUTs must be in a steady state for evaluation to satisfy the basic condition - quiescent current. Therefore, any pull-up or pull-down logic, and RAM sense amplifiers in the design, can greatly decrease the fault detection sensitivity.

Another downside of the I_{DDQ} testing is that it is time consuming since accurate results can not be obtained until the circuit is settled to its steady state after the input stimulus is applied. Consequently, this technique is not a good applicant for at-speed testing.

It is difficult to set the I_{DDQ} current limitation to a certain value for all DUTs as the I_{DDQ} threshold can vary depending upon the design and process. In addition, high

precision measurements are required for I_{DDQ} testing since the leakage current is very low, therefore, the signal to noise ratio (SNR) is very low as well.

I_{DDQ} Test Models and Methods

There are many models used in I_{DDQ} to detect a variety of faults, here are some models often seen in the test.

- Stuck-at fault model: to detect a fault by forcing the opposite value of stuck-at fault at defective node. Using this model is able to detect bridging fault (Rajsuman2000).
- Transistor short model: A certain pattern of inputs are applied for detecting shorts between the terminals.
- Bridging fault model: Forcing two opposite values at the two nodes in order to detect shorts between these two adjacent nodes.

I_{DDQ} Test Methods

Different I_{DDQ} methods along with other testing techniques, such as voltage-based measurements, are used in order to achieve the optimal results and minimize their disadvantages. The target applicant for “every vector I_{DDQ} ” testing, which will perform I_{DDQ} test after every vector is applied to the input, is normally a prototype device. In other cases, “selective I_{DDQ} ” testing, which only uses a portion of the vectors instead of every one, is an effective method to detect the majority of the faults. “Supplemental I_{DDQ} ” test is used, in addition to voltage based testing techniques, when “full-speed functional tests” are conducted.

I_{DDQ} Design Concerns

Some of the important factors, which will determine I_{DDQ} testability, need to be considered when designing an I_{DDQ} test are:

- Contention current should be avoided in the design since the test is based on the quiescent current. Moreover, no floating nodes should be present in the testing circuit.
- If current sources or sinks and sense amps are necessary for the circuit, then the use of a different power supply from the testing circuit may prevent them from affecting the testability.
- Any dynamic nodes should be fully charged or discharged on each clock cycle.
- A switch or controller is highly recommended for shutting any unnecessary static current (Rajsuman2000).
- Disable any free running oscillators during the test as they can draw a certain amount of the current.

I_{DDQ} Challenges and Future Trends

Setting the I_{DDQ} threshold becomes more challenging as transistor size shrinks smaller and smaller and it also can be a factor in causing the I_{DDQ} values to vary. Therefore, it is critical to identify the elements that will cause I_{DDQ} to fluctuate. For example, temperature, voltage, and other parameters can vary the I_{DDQ} values.

The testing time can be long since a period of time is needed before the circuit settles into a steady state. I_{DDQ} testing remains problematic if a circuit under test contains non-CMOS devices since it is designed based on CMOS leakage current.

There are many areas in I_{DDQ} testing that still need to be improved, particularly, the delay fault and using I_{DDQ} as an analysis tool.

BUILT-IN-SELF-TEST (BIST)

With memory size on the increase, the complexity and testing time rise as well. This means that the cost of testing goes up and, as a result, the products overall costs rise too. It becomes more and more challenging to perform the necessary tests outside the circuit when there is no physical pin to access the integrated circuit memory, for example, circuits in *embedded* memory. To ensure those memories' quality and reliability, a built-in-self-test (BIST) circuit is added internally to test the circuit. In addition to detecting faults and defects, BIST is able to supply the failure data for the defect types and their locations for analysis and repair. This information is important because the testing results can help determine if the current process has problem or if the defects can be eliminated in the future.

BIST Advantages and Disadvantages

There are many advantages for using BIST. The testing cost can be reduced because it does not require the expensive automatic testing equipment (ATE). The fault coverage increases since it allows multiple testing -- internal and external tests are performed at the same time. The on-line BIST allows performing the test while the normal operation is in process or conducts off-line BIST when the system is under the diagnosis tests. Testing time, one of the key contributors to cost, will be reduced due to the possibility of running the test simultaneously if the memory is divided into several banks. Since the memory chips are also integrated onto boards and systems, BIST can be conducted at each level.

The benefit of using BIST in high-density memories can be extended if it is associated with Built-in-self-repair (BISR). The reason for this is whenever random failures and their locations are found in a bit, BISR is able to replace the defective bit with spare rows or columns. However, as we expect, if the failure rate is high, BISR is no longer practical due to the large amount of extra rows and columns needed.

According to Pateras, the Director of Engineering for manufacturing test software at LogicVision Inc., the BIST has some advanced benefits compared to ATPG techniques. ATPG will normally generate a larger test vector and pin-to-pin timing can be very critical. On the other hand, BIST does not need to store testing data, and the timing restriction is insignificant [Pateras2002].

However, the benefit for adding BIST to a small memory array is insignificant in relationship to the high percentage of the chip area usage. Because of this, the BIST can seriously degrade the memory circuits operation if the controller size and memory array size are comparable [Boyer2003]. Boyer also indicated, that "all memory input pins have a multiplexer to select between BIST and system signals and that can lead to routing congestion." Therefore, the best applicants for adding BIST should be to those circuits having large memory size with a minimum number of ports [Boyer2003].

Other downsides of the BIST are revealed when the memory density is reduced due to the added circuit; the circuit design will be more complicated as the additional circuit also requires extra pins to connect other circuit; the testing speed is possibly reduced due to the longer testing path and higher overhead of the circuit.

Testing Types and Algorithms

Testing Types

- Exhaustive testing

Exhaustive testing can be time consuming since it needs to test all 2^N logic values if the device has N inputs. The benefit of exhaustive testing is that it is capable of discovering all detectable faults since all possible combinations of the input stimulus are applied. On the other hand, it is not practical to conduct exhaustive testing if N is a large number (>20) due to the enormous test vectors and the matter of the lengthy test time.

- Pseudo-exhaustive testing

The benefit of the pseudo-exhaustive testing is that it uses fewer test patterns to achieve exhaustive testing results. The overhead is lower than exhaustive testing.

- Pseudorandom testing

Test patterns are less than 2^N , or are partial of exhaustive testing. The characteristic of pseudorandom is the mixture of random and deterministic. Linear feedback shift registers (LFSRs) are used to generate the test patterns.

Testing Algorithms

March Tests

March test is when a set of test instruction steps (normally is generated by LFSRs) are performed to test each cell of memory array in order. It is not critical as to where the test is starting, at the bottom or at the beginning of the memory array, as long as a

complete test is conducted for each cell before moving to the next one. The test usually takes 4 to 17 ns to finish per cell [BIST of Memory]. The March test is commonly used to detect stuck-at faults in both RAM and address decoders on top of all transition faults and some coupling faults [Otterstedt98].

Neighborhood Pattern Sensitive Tests (NPSF):

“A NPSF tests every cell of the memory in relation to its set of 5 or 9 neighboring cells (including the base cell)”. This test takes 195 ns to be completed [BIST of Memory], longer than the march test.

BIST Applications

BIST is widely applied in various memory circuitries: SRAMs [Tehranipour2000], DRAM, flash [Yeh2002], and embedded memories.

Using BIST in SRAM testing was approved effective in Tehranipour and Zavabi’s paper. They implemented a so called “length 9N test algorithm” to conduct March testing on SRAM. Their testing method was able to utilize the existing hardware and software and the results showed an achievement of 100% fault coverage without increasing any overhead [Tehranipour2002].

Using BIST for testing embedded memories also showed great advantages simply because the memory is very compacted, consequently, the BIST overhead is small in comparison to the overall memory circuit [Burgess2000].

Figure 12 shows the block diagram of memory with BIST. The BIST circuitry is enlarged for explanation purpose. Once a failure is flagged, its address will be scanned out for diagnosis [Burgess2000].

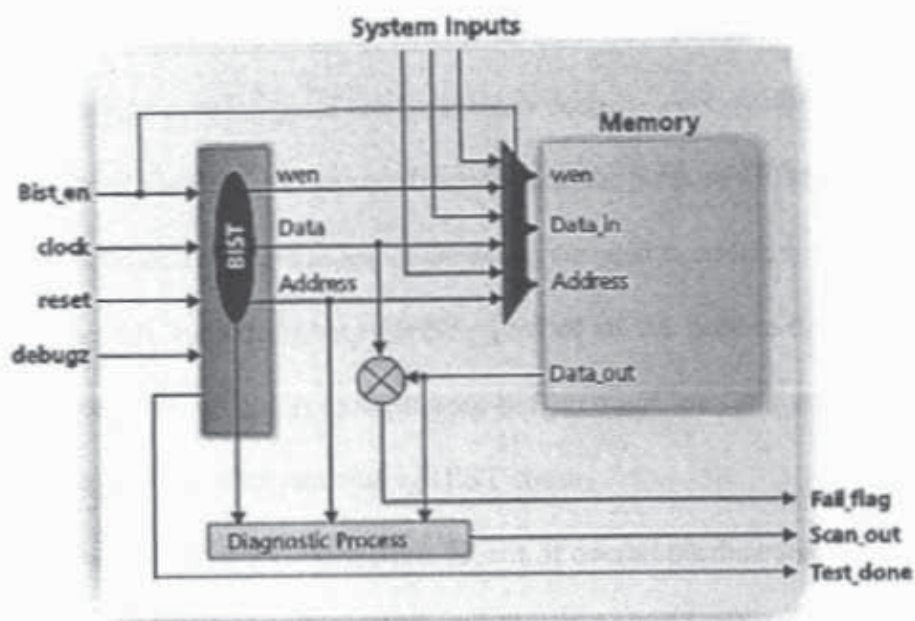


Figure 12 Memory with BIST block diagram (Ref: [Burgess2000])

BIST Design Concerns

There are many different designs in which to conduct a BIST. The following are some concerns when designing a BIST so that each test will meet specific goals.

- **Faults:** We would like to know what faults the test should detect. In Otterstedt and Niggermeyer's paper, they used modified march test to detect address faults, stuck-at faults, coupling faults, and transition faults [Otterstedt98]. The NPSF can detect "the class of Active, Passive, and Static Neighborhood Pattern sensitive faults, which including stuck-at faults and all coupling transitional faults" [BIST of Memory].
- **Time:** Timing is always a big issue because the goal is to reduce the test time to a minimum. The bottom line is how much more time is allowed before it slows down the device significantly.

- **Area:** How much area should we sacrifice to gain the BIST benefits? For instance, the small memory arrays normally do not justify the benefits of the BIST because their high percentage of area usage can decrease core circuit's performance. The sizes of the BIST are varied depending on the purpose of the design.
- **Overhead:** There are at least two issues we need to consider for minimizing overhead when designing a BIST. First of all, the need to determine which method is required in order to keep the overhead low, and secondly, what type of memory is suitable for adding BIST circuit.
- **Cost:** Since testing expenditure is part of overall product cost, it is critical to design a test method that is economic but has high performance. Consequently, the trade offs between the testing time, the fault coverage, testing method complexity of each test methods and their costs are need to be considered as well at the designing stage.
- **Others:** More advanced techniques, such as Automatic Built-in Self-Test (ABIST) has been developed based on IEEE Std 1149.1 and its extensions. The goal for testing in the near future is to apply Electronic Systems Test Automation (ESTA) when design a Hierarchical and Integrated BIST (HIBIST) where it is possible to test the chips all the way to the systems.

SUMMARY

The intent for this paper is to introduce various testing concepts and to discuss some of the current testing techniques and their set of trade offs. In addition, some of the design concerns are addressed in order to deal with different testing needs.

There are many advantages of using the boundary scan testing technique. The boundary scan enabled testing the same device at different levels including: components, PCBs, and systems. Further, this method is also able to enhance and support other testing features and makes the testing more efficient. However, the overhead can become the limiting factor. In general, the design goal is to decrease the overhead and increase chip usage area while maintain high fault coverage.

I_{DDQ} test showed its unique advantages—detect some of the defects, while the voltage based measurement method does not have the same capability. The limitation when using I_{DDQ} is that any dynamic circuit will reduce the defect detection sensitivity. Since I_{DDQ} test is designed for CMOS circuit only, it becomes useless when non-CMOS circuits exist in the DUTs.

Using BIST has been proven beneficial in embedded memory circuits, high density DRAM and other large memory circuits. Moreover, BIST reduces the use of expensive external ATE. However, when BIST is not designed properly, especially, in the small memory circuit, the advantages disappear. The problem is that extra overhead causes complexity, as result resulting in low yield. Therefore, a BIST design that can achieve maximum fault coverage, minimum overhead, and more flexibility are strongly preferred.

REFERENCE

- [1149.1-2001] IEEE Computer Society. IEEE Standard Test Access Port and Boundary-Scan Architecture. IEEE Std 1149.1-2001. The Institute of Electrical and Electronics Engineers, Inc. New York, NY Jul. 23, 2001.
- [BIST of Memory] Built-In Self Test of DRAM Chips: An Overview of Issues and Papers Sept. 11, 2003 <http://www.eecs.harvard.edu/cs245/papers/Davidb.html>
- [Boyer2003] Boyer, Jeff and Ron Press. New Methods Test Small Memory Arrays. Test & Measurement World. Feb. 1, 2003. Aug. 25, 2003. <http://www.reed-electronics.com/tmworld/index.asp?layout=article&articleid=CA274065&rid=0&rme=0&cfid=1>
- [Boundary Scan] Boundary Scan /IEEE Standard 1149.1 Aug. 25, 2003. <http://toolbox.xilinx.com/docsan/xilinx4/data/docs/pac/appendixa2.html>
- [Burgess2000] Burgess, Ian. Test and Diagnosis of Embedded Memory Using BIST. Evaluation Engineering. 2000. Sept. 26, 2003 <http://www.evaluationengineering.com/archive/articles/0300mem.htm>
- [Chen2001] Chen, John T. Janusz Rajski, , *et al.* Enabling Embedded Memory Diagnosis via Test Response Compression Proceeding of the IEEE VLSI Test Symposium Apr.29-May 3, 2001 p292-298
- [Crouch] Crouch, Al. Semiconductor IC Test and Design-for Test Fundamentals Aug. 25, 2003. <http://www.inovys.com/login/DFTwhitepaper.pdf>
- [Gopel electronic] Gopel Electronic. IEEE Std 1149.1(Boundary scan) Tutorial Sep. 6, 2003 <http://www.goepel.com/eng/bsc/tutorialfr.htm>
- [Isern & Figueras 95] Isern, Eugeni and Joan Figueras. IDDQ Test and Diagnosis of CMOS Circuits. IEEE Design & Test of Computers 1995. pp60-67
- [Klenke98] Klenke, Bob. Test Technology Overview Module 43 Aug. 30, 1998. Aug.25, 2003 http://www.cedcc.psu.edu/ee497f/rassp_43
- [Niggemeyer2001] Niggemeyer, Dirk and Elizabeth M. Rudnick. Automatic Generation of Diagnostic March Tests. Proceedings of the 19th IEEE VLSI Test Symposium (VTS '01) 2001. p0299. Computer Society

- [Otterstedt 98] Otterstedt, J., and D. Niggemeyer. Detection of CMOS address decoder open faults with march and pseudo random memory tests. Proceedings of International Test Conference Oct.18-21, 1998. pp53-62. ISSN/ISBN: 1089-3539.
- [Peteras2002] Peteras, Stephen. BIST Versus ATPG - Separating Myths From Reality EEdesign Nov. 27, 2002. Sept. 17, 2003
<http://www.eedesign.com/features/exclusive/OEG20021127S0040>
- [Rajsuman2000] Rajsuman, Rochit. Iddq Testing for CMOS VLSI. Proceedings of The IEEE, Vol. 88, No. 4, Apr. 2000 p544-566
- [Sidorowicz2002] Sidorowicz, Piotr.R. and Janusz A. Brzozowski,. A Framework for Testing Special-Purpose Memories. IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems: IEEE Circuits And Systems Society.: Volume: 21 Issue: 12: Dec.2002 p1459-1468
- [Tehranipour2000] Tehranipour, M. H., and Z. Navabi. Zero-overhead BIST for Internal-SRAM Testing. Proceedings of the 12th International Conference on Microelectronics (ICM 2000): Oct. 31-Nov. 2, 2000. pp109-112
- [Wondolowski99] Wondolowski, Mike, Ben Bennetts, and Adam Ley. Boundary Scan: The Internet of Test IEEE Design & Test of Computer Vol16, No.3, July-September 1999, pp34-43
- [Yeh2002] Yeh, Jen-Chieh, *et al.* Flash Memory Built-In Self-Test Using March-Like Algorithms Proceedings of the First IEEE International Workshop on Electronic Design, Test and Applications (DELTA.02). p137. Computer Society. 2002