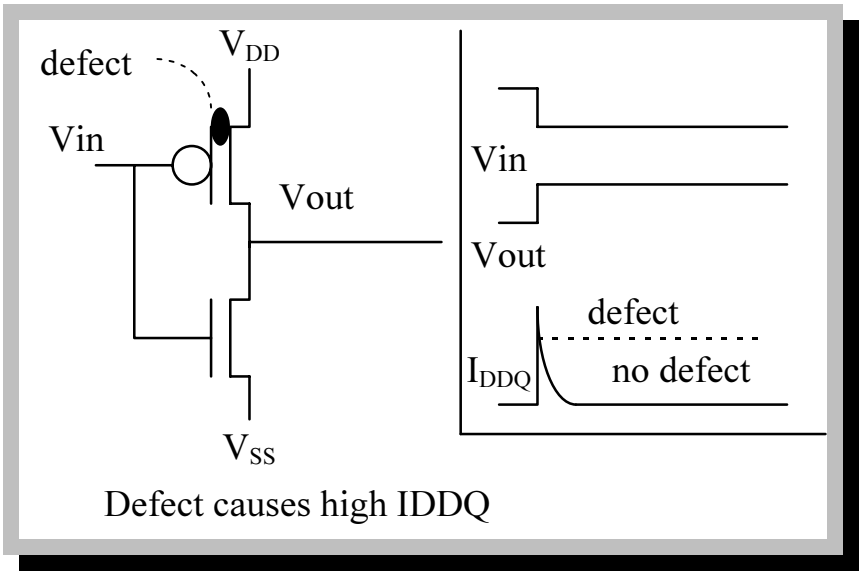
IDDQ Made Easy

CMOS IDDQ Test Methodology

Fundamental Concepts



SYSTEMS SCIENCE

First Edition, June 3, 1996
Revised January 2, 1997

Systems Science, Inc.
1860 Embarcadero Road
Suite 260
Palo Alto, CA 94303

(415) 812-1800 (Voice)
(415) 812-1820 (Fax)
info@systems.com | iddq@systems.com
http://www.systems.com

IDDQ Made Easy

CMOS IDDQ Test Methodology

Fundamental Concepts

Bob Duell
Systems Science Inc.
1860 Embarcadero Road, Suite 260
Palo Alto, CA 94303
January 2, 1997

1. What is IDDQ?

I_{DDQ} is the IEEE symbol for the quiescent power supply current in a MOS circuit. I_{DDQ} (or IDDQ for our purposes) refers to the quiescent current in CMOS integrated circuits.

IDDQ testing is a cost-effective test strategy for digital CMOS ICs with the potential to improve the average quality level of your IC production with little extra effort or cost. The purpose of this booklet is to explain IDDQ testing, its benefits, and how to use it to improve CMOS IC quality and reliability.

2. Review of Common Digital Test Strategies

To fully understand the impact that IDDQ testing has on final product quality, a review of some test program strategies used for ATE (Automatic Test Equipment) test program development is helpful.

2.1. Test Program Development

Test program development encompasses the planning, engineering, and implementation of a test strategy resulting in a test program that assures satisfactory product quality level. There are three basic test development strategies that are often combined.

2.1.1. Functional Test Development

Functional test development is usually performed by the circuit designer originating with the design verification simulation. The purpose of the functional test is to cause the design to behave the same way that the designer expects that the end user will use it. Logic simulation data are translated to ATE format for use by the ATE. Functional tests are usually created by capturing the values of a logic simulation at the primary I/O pins. They can also be created directly in the ATE's tester language without any logic simulation, although this is rarely done. Fault simulation is optional.

2.1.2. Structural Test Development

Structural test development relies on a computerized model of logical circuit faults. The goal of these tests is to detect as many of the modeled faults as possible without regard for the functionality of the circuit. A fault simulator grades the effectiveness of the test's stimulus. The most prevalent fault model is the "single stuck-at" model (each circuit node is modeled as being connected to a constant '1' or '0'). It assumes that most circuit defects are detected by test sets that detect a high percentage of these faults. Other fault models include: bridging faults, stuck-open faults, delay faults, and leakage faults. These tests are implemented with "Boundary-scan", "Full-scan", "Partial-scan", and "Non-scan" testability methodologies, and often begin with functional test sets that are enhanced with manually, algorithmically or automatically (ATPG) generated test patterns. Since these fault models do not represent all of the possible circuit defects, some devices pass these tests but fail in actual use. Structural test development is normally performed by Design Test Engineers in collaboration with Design Engineers. Test sets are translated to the ATE's language from fault graded logic simulations of the design.

Most of today's test development is structural because the fault simulator's report provides a means to quantify the effectiveness of the test. Structural test development usually begins with functional logic simulations that are fault graded and enhanced for higher fault coverage. The graded simulation results are converted to make Functional Test Sets (the ATE stimulates the circuit inputs and monitors the voltage on the circuit's output pins) and/or IDDQ Test Sets (the ATE stimulates the circuit inputs and monitors the IDDQ power supply current).

2.1.3. Physical Defect Test Development

Physical defect test development consists of listing possible real physical defects that occur in a circuit, then creating specific tests designed to detect each defect.

These tests are normally developed to cover failures of devices that have passed test but fail in service. The failing device is analyzed and the defect is identified. While this is the only sure method to guarantee zero defects, it is unlikely that all of the possible physical defects will be identified. Fortunately, IDDQ tests often detect faults that are not predicted (modeled or simulated) thereby reducing the amount of post-mortem analysis required to maintain a low defect level.

2.2. ATE Test Types

2.2.1. Comparison Tests

Comparison tests compare the DUT (Device Under Test) to a "known good" reference device. Two problems with this technique are obvious: 1. how do you prove that the reference device is good, 2. how do you know that you have provided adequate stimulus to completely exercise the reference device? Comparison tests are rarely used in modern test programs.

2.2.2. Functional Tests (a.k.a. Voltage Tests)

The word "functional", when used to refer to ATE Test Programs, describes a stimulus/response test program in the language of the target ATE. This differs from the use of the term when it describes Test Development (a simulation performed to simulate the functionality of the DUT without regard to its structure). The tester's software uses a Functional Test Set to drive the DUT's inputs and compare its output pin voltages to the test set's stored values. The tester's pins are wired to the DUT, and the tester software maps the tester's pins to the DUT's I/O pins. These test sets are derived from Functional and Structural Test Development logic simulations.

2.2.2.1. Scan

Scan tests are a variation of functional tests. They require a procedure in the ATE's test set to read from and write to the DUT's scanned flip-flops and latches.

Scan is a design-for-testability methodology that modifies the design's flip-flops and latches to include a test mode that enables test signals to be loaded, or latched data to be read, directly by the test program. Scanned sequential elements are "stitched" together to form a serial shift register when in test mode.

Scan makes sequential circuits more observable and controllable from the ATE. Full-scan simplifies ATPG (automatic test pattern generator) program operation by

replacing sequential devices with scan I/O, thereby allowing the ATPG to operate as if only combinational cells are present.

2.2.2.2. Boundary-scan (JTAG - IEEE 1149.1)

Boundary-scan methodology treats the scan latches that surround a sub-circuit partition as tester I/O. Test sets can be developed that test each partition separately and partition-to-partition connectivity. Each sub-circuit's boundary scan latches are accessed by the tester through a TAP (Test Access Port) controller.

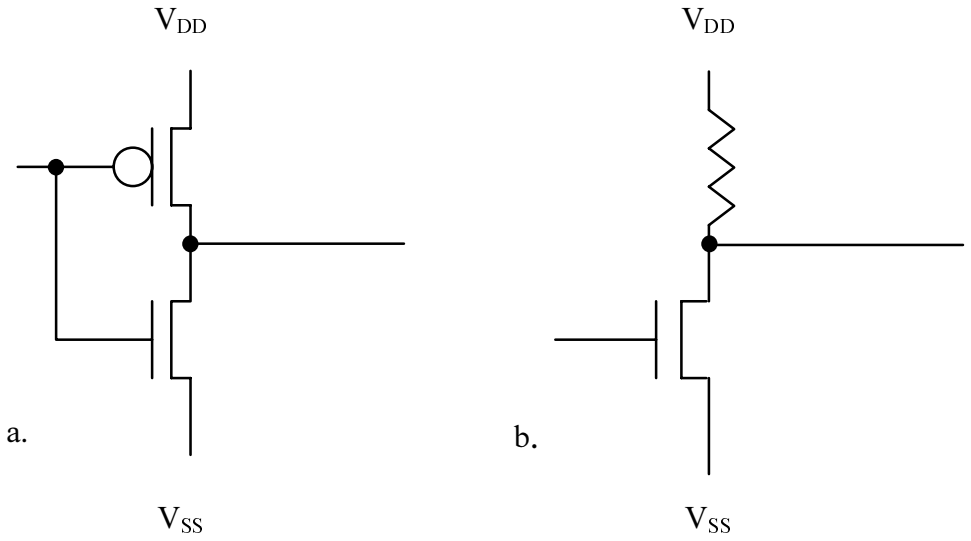
2.2.3. IDDQ Tests

Normally, IDDQ testing on ATE consists of a functional test set, with or without scan, to stimulate the DUT. At test steps predetermined to be suitable for IDDQ tests, the tester is halted, and the DUT's power supply current is measured by the ATE. The resulting value is compared to a reference value, for a go/no-go test response. IDDQ test methodology is the most successful method for detecting CMOS physical device defects.

3. IDDQ Test Concept

IDDQ testing is based on the premise that CMOS circuits draw extremely low leakage current when no transistors are switching. The ideal IDDQ testable circuit is a fully complementary and fully static CMOS design. The concept of using complementary p-MOSFET and n-MOSFET transistor pairs in the CMOS configuration was designed expressly because no current is consumed by the circuit except when switching. This attribute of CMOS implies 100% IDDQ testability by its very definition.

In practice, designs that are not ideal are IDDQ tested, but deviation from ideal CMOS always has a negative impact on the maximum IDDQ test coverage that is possible for the design. Even so, IDDQ tests have proven effective in detecting many faults that otherwise would remain undetected.

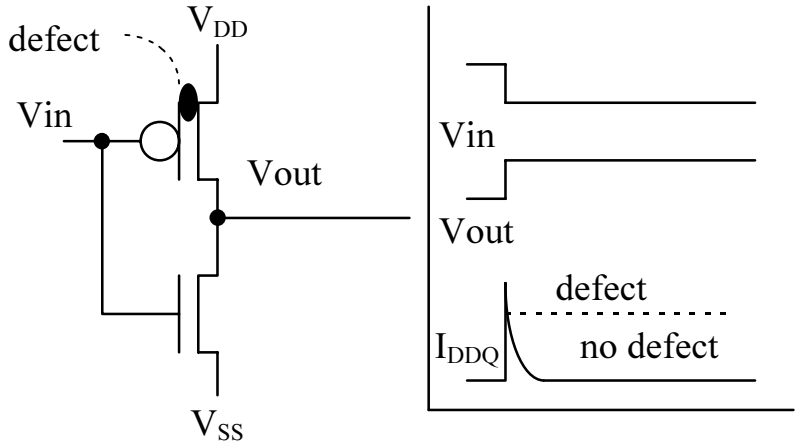


Fully complementary CMOS Non-complementary - R in place of p-MOSFET

Figure 1

3.1. How Does Measuring IDDQ Detect CMOS Defects?

A CMOS integrated circuit in quiescent state typically draws less than 10 micro amps. Any defect that causes higher IDDQ than the assumed threshold value can be detected by monitoring IDDQ.



Defect causes high IDDQ

Figure 2

3.2. IDDQ Test Attributes

3.2.1. Every trace is observable

IDDQ tests differ from functional tests in that they sense current rather than voltage. This feature enables the DUT's power supply pins to serve as ultra sensitive test observation points capable of sensing every circuit trace and via. All IDDQ faults propagate to the power supply.

Functional tests sense voltage, and are sometimes refered to as "Voltage Tests". Voltage is tested at the DUT's output pins. Faults must propagate to the DUT's output pins for the tester to differentiate between a good or bad DUT.

3.2.2. High fault coverage with few test vectors

Normally, over 50% of a circuit can be tested with one or two IDDQ test vectors. Figure 3 illustrates that the first few vectors detect the majority of faults.

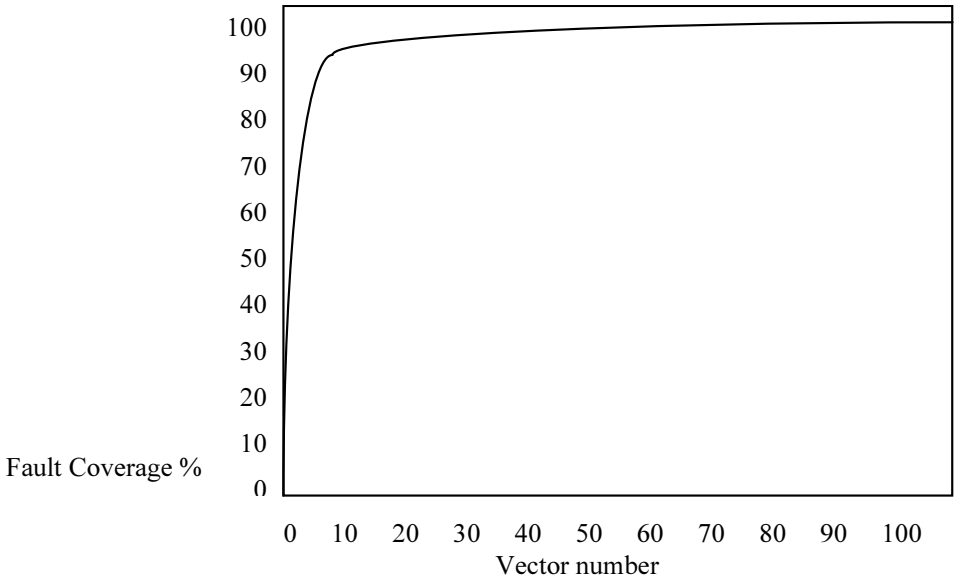


Figure 3 Typical IDDQ Fault Coverage vs. # of Test Vectors

Derived from: Maxwell, P.C., R.C. Aitken, V.Johansen, I. Chiang, "The Effectiveness of IDDQ Functional and Scan Tests: How Many Fault Coverages Do We Need?", Proc. 1992 International Test Conf., pp. 167, Figure 4, and modified to average fault model types.

3.2.2.1. IDDQ Fault Behavior

A paper presented at ITC96 (T. Powell, et. al., "Correlating Defects to Functional and IDDQ Tests", Proc. 1996 International Test Conference, pp 501-510) describes a TI study and concludes:

"IDDQ defects can be classified at two types: IDDQ pattern sensitive types and IDDQ pattern insensitive types which means that any current measurement pattern will screen the defect. By far the majority of IDDQ defects are the IDDQ pattern insensitive type. Gate oxide defects tend to be pattern sensitive defects while the metal shorts to GND or VDD tend to be pattern insensitive defects.

When selecting IDDQ tests to apply, it is most important to target the pattern sensitive type since the other type will be screened no matter which tests are applied. The method for selecting the patterns, whether it is a pseudo stuck-at-fault current grade or a transistor short model, did not matter in selecting the more effective tests. Both types were equally effective although the grades given were slightly different. It is important, however, to use an IDDQ fault grading tool to select the critical patterns to effectively screen the largest number of defects. ..."

Systems Science's PowerFault-IDDQ is especially well suited for finding the "pattern sensitive" fault types since it optimizes test vector selection for maximum fault coverage over single or multiple test benches. The above graph indicates that many faults are covered in the first few vectors. These abundant first faults can be detected with most of the qualified IDDQ test vectors, whereas the faults found on the right side of the graph require specific test vectors to detect specific faults. PowerFault-IDDQ's cell-boundary fault model is especially well suited to insure optimal fault coverage.

3.2.3. Defects Detectable by IDDQ Tests

3.2.3.1. Node Bridges

When a bridge connects two nodes that are driven by conflicting voltages, after passing through one logic gate, propagates a strong logic strength. High IDDQ occurs in at least one of the logic states. Node bridges are detected by setting the adjacent nodes to opposite logic states. These faults are easily detected by functional test sets provided that the nodes are easily observable and the bridge is low impedance.

3.2.3.2. Gate Oxide Shorts

Gate oxide shorts usually cause weakened logic voltages, but they do not generally cause low frequency logic errors. These faults are likely to pass functional tests, but may cause the device to fail in service as it ages. Faults are detected by placing the gate's input node in both '1' and '0' states.

3.2.3.3. Leaky pn Junctions

Leaky pn junctions (drain or source to bulk/well leaks) are rarely detected by functional tests, but are easily detected by IDDQ tests

3.2.3.4. Power Supply Bridges

Most power supply bridge faults are detected by both functional and IDDQ tests.

3.2.3.5. Punchthrough

Punchthrough (drain to source leaks) are seldom detected by functional tests, but they are easily detected by IDDQ tests.

3.2.3.6. Parasitic Leaks

Parasitic leaks are seldom detected by functional tests, but they are readily detected by IDDQ tests.

3.2.3.7. Open Transistor Gates, Drain and Sources

Open gates often appear as timing faults. Drain and source opens can cause a combinational circuit to exhibit memory characteristics. Open transistor element faults may or may not cause elevated IDDQ levels.

3.2.4. Defects Untestable by IDDQ Tests

IDDQ tests are not reliable for open transistor faults.

3.2.5. Defects That Are Untestable by Functional Tests

3.2.5.1. Redundant Logic

Functional tests cannot discern among redundant logic elements since they are constrained to observing outputs common to the redundant logic. IDDQ tests observe redundant logic elements independently, without regard to their common output.

3.2.5.2. Shorts

Resistive shorts are especially difficult for functional tests to detect but are quite susceptive to detection by IDDQ tests. Low impedance shorts are normally detectable by either method.

3.2.5.3. Burn-in Failures

IDDQ tests have proved to be an effective pre-screen for burn-in failures. Many devices that pass functional tests but fail IDDQ tests also fail burn-in tests. IDDQ tests are not generally recognized as a replacement for burn-in tests, but are useful in improving burn-in test throughput, and in reducing post burn-in failures. However, a paper (T. Henry, T. Soo, "Burn-in Elimination of a High Volume Microprocessor Using IDDQ", Proc. 1996 International Test Conference, pp242-249) offers a technical and economic case study by INTEL, where burn-in tests are eliminated by IDDQ testing.

3.2.5.4. Delay Faults

Delay or timing failures frequently originate with faults that are detectable by IDDQ tests. While IDDQ tests do not detect all delay faults, they can significantly improve the odds that such faults are detected.

IDDQ tests along with very low voltage functional tests (J. Chang, E. McCluskey, "Detecting Delay Flaws by Very-Low-Voltage Testing", Proc. 1996 International Test Conference, pp367-376) can detect most delay faults without specific delay fault modeling.

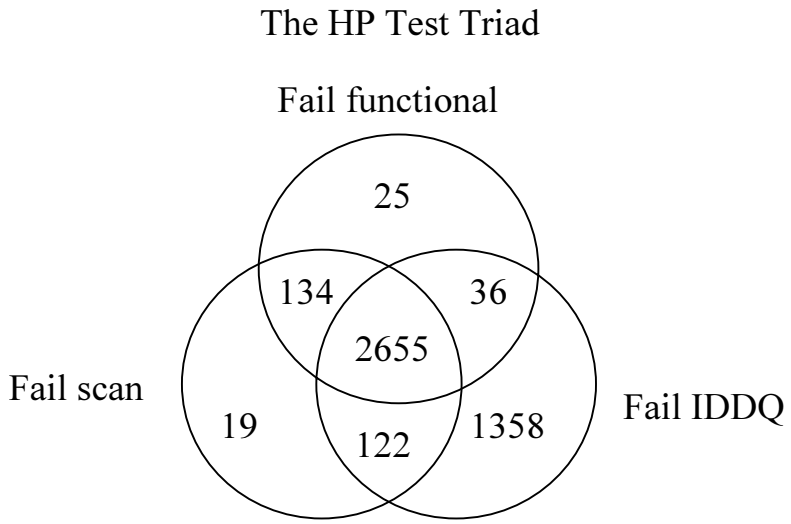
3.2.5.5. Potential Detects

A potentially or partially detected fault is defined in the context of the traditional stuck-at fault simulation. It refers to the case where a fault free node is a known '1' or '0' value but the faulted node is 'X' (unknown). IDDQ tests cannot generate potentialy detectable faults, moreover, they solidly detect those that are generated by traditional fault simulators.

---

4. Relative Effectiveness of the Test Types

Figure 4 illustrates the published results of a study performed at Hewlett-Packard where three wafer lots consisting of 26,415 die were tested and binned. It graphically illustrates the relationship among functional tests, scan tests, and IDDQ tests. It shows that each test type is capable of detecting faults undetectable by other means. Although not the specific intent of the paper, this graphically illustrates that the advantage of IDDQ testing is too important to ignore.



Maxwell, P.C., R.C. Aitken, V.Johansen, I. Chiang, "The Effectiveness of IDDQ Functional and Scan Tests: How Many Fault Coverages Do We Need?", Proc. 1992 International Test Conf., pp. 168-177

Figure 4

4.1. Comparison of Test Method Success for CMOS ICs

	P = Poor	F = Fair	G = Good
Defect	Functional	Structural	IDDQ
Gate Oxide Short	P	P	G
Bridge	F	F-G	G
Parasitics	P	P	G
pn leakage	P	P	G
Punchthrough	F	F	G
Open drain	P	F-G	F
Open gate	F	G	F

Table 1

Condensed from J.M. Soden and C.F. Hawkins "Current Monitoring (IDDQ Testing) for Efficient Detection of CMOS IC Defects and Faults", A Texas Instruments Presentation, March 11, 1993, Houston, TX

5. Efficient Test Development

5.1. Simplify Test Development by Developing IDDQ Tests First

5.1.1. Start With Design Verification Test Suite

Use the functional test or design verification test suite for initial IDDQ test development. This test suite usually stimulates the design thoroughly with fewer test vectors than test suites that target specific faults.

5.1.1.1. Augment design verification tests to toggle all nodes

It is a good idea to run the logic simulator's toggle test to verify that the design verification test suite has exercised all of the design's nodes. While complete toggle coverage does not assure that design verification is complete, it is a good starting point for both design verification and for the following DUT test development.

5.1.1.2. Minimize the time that bi-directional pads are at Z

Drive bi-directional pads from inputs, or insure that they are driven to a known value. This minimizes contention problems and insures greater success in later test development activities.

5.1.1.3. Select the maximum allowable number of IDDQ test vectors

5.1.1.3.1. IDDQ test vector optimization

Due to the relatively long time period required for a stable high precision IDDQ current measurement to be made, usually only a few IDDQ test vectors are allowed. Therefore, it is crucial that the selected vectors cover as many IDDQ faults as is possible. Most IDDQ vector selection tools select their vectors on the basis of the percentage of remaining undetected faults detected by each vector candidate. More advanced tools, such as System Science's PowerFault-IDDQ™, actually optimize by analyzing the coverage of each of the vector candidates to select the vector set with the highest possible overall fault coverage. PowerFault-IDDQ can optimize vector selection over single or multiple test benches.

5.1.1.3.2. For IDDQ, ATPG alone may be ineffective

When only a few IDDQ test vectors are allowed, it is important that there be as much IDDQ fault activity with as few vectors as possible. ATPGs are "fault-

directed", meaning that the ATPG attempts to generate a test vector sequence for each fault that it encounters. This methodology may create sparse test sets (low fault activity per test pattern).

Some ATPGs can generate random test patterns. While for some designs, random stimulus yields good results, random test stimulus generation can be accomplished easily without expensive ATPG programs.

5.1.1.4. Remove detected stuck-at faults from subsequent fault simulation

Functional/structural tests are usually based on the "stuck-at" fault model. IDDQ "stuck-at" faults are a superset of structural "stuck-at" faults. IDDQ tests detect high impedance defects that functional tests do not. However, low impedance stuck-at defects are detected by both methods. Therefore, it is valid to remove the stuck-at faults detected by IDDQ tests from consideration in subsequent fault simulations.

5.2. Develop and Enhance Functional Tests

Use a combination of manual, algorithmic, or ATPG, generated test vectors to achieve the desired "stuck-at" fault coverage.

5.2.1. The Stuck-at Fault Simulator

With the IDDQ stuck-at faults removed, the fault simulator's workload is reduced. Fault simulation workload can be further reduced if it is possible to use unit delay simulation (results of unit delay and full timing simulations are equal).

6. IDDQ Test Measurement Strategies

Before IDDQ tests can be implemented on ATE, a threshold pass/fail reference value must be established. This value represents the quiescent IDD value for the device. Once this value is established, there are several methods of performing the actual IDDQ measurements and comparing them to the reference value.

6.1. Threshold Determination

One of the most controversial aspects of IDDQ testing is determining the pass/fail reference value that will be used to test an IC. One method is to sample representative ICs to arrive at a practical threshold value. Another method is to run SPICE to determine leakage values for each cell type then multiply those values by the

number of each cell type in the device. Yet another method is to simply guess based on previous experience.

Typical reference values are between 1 and 20 micro-amps. The risk of setting this value too low is that possibly good devices may be rejected. The risk of setting the threshold value too high is that bad devices escape detection. Some devices will fail IDDQ but pass functional tests.

6.2. Off-Chip Current Monitoring

Slow test times are a real disadvantage to the IDDQ test strategy. To partially overcome this problem, some test facilities have devised IDDQ measuring and threshold detecting hardware that allows the ATE to perform IDDQ tests at higher speeds. These hardware implementations usually consist of differential amplifiers with threshold reference circuitry, all of which is mounted on the tester's load board. One such example is explained and illustrated in the Hewlett-Packard Application Note 398-3 "Measuring CMOS Quiescent Power Supply Current with the HP 82000". (Figure 5 and Figure 6)

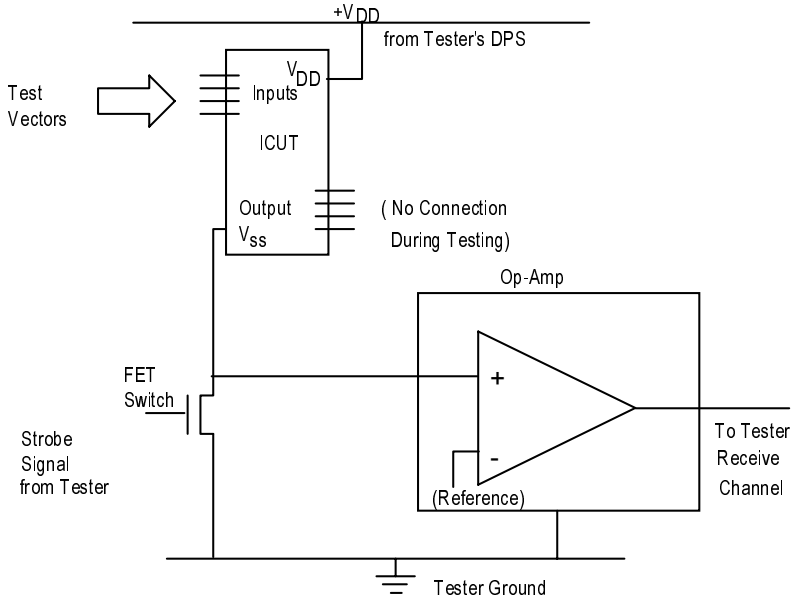


Figure 5 Test Set Up for Off Chip Current Monitor

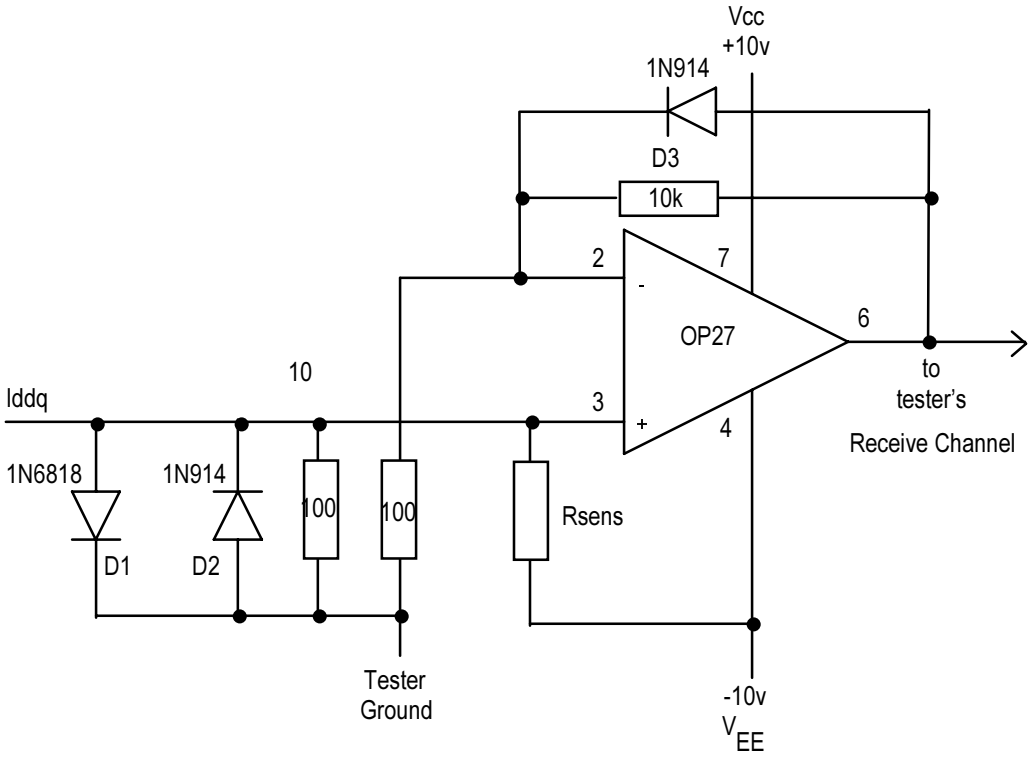


Figure 6 Op-Amp Circuit

6.2.1. QTAG

The QTAG (Quality Test Action Group) began at the ITC '93 (International Test Conference). It is supported by both the semiconductor industry and the academic community. It is charged with developing a standard method for IDDQ testing. QTAG is developing a standard off-chip current monitoring device. Documents relating to the activities of QTAG can be found in the proceedings of the ITC.

6.3. BICS

Built-in current sensing circuits have been described in several ITC papers. While the idea is attractive, there are several technical issues that are unresolved in its implementation (such as crosstalk, routing, etc.). No commercial applications of this technology are noted at the present time.

6.4. PMU

The tester's parameteric measuring unit is by far the most common tnstrument for measuring IDDQ. It has the advantage of being readily available to test programmers without modifing hardware.

7. IDDQ Test Program Considerations

7.1. Select Functional Vectors that Qualify for IDDQ

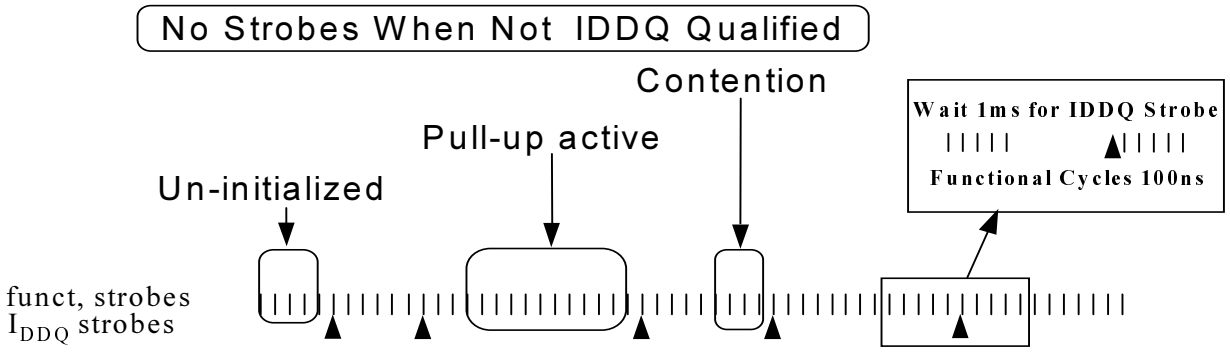


Figure 7 Functional Vectors with IDDQ Interspersed

Determine which functional test vectors qualify for IDDQ. This can be done manually on the tester by stopping to measure IDDQ at each functional test vector. This method is very useful to determine if the DUT is IDDQ testable. If analog circuits or RAM currents cannot be disabled, there will be no vectors that exhibit low current. It is also possible to create IDDQ test sets in this manner, but when the number of IDDQ vectors is limited, it is wasteful in terms of test coverage.

IDDQ test generator tools fault grade each IDDQ test vector. Besides the confidence instilled by having test coverage numbers, fault grading provides the possibility of optimizing vector selection for higher test coverage. Advanced tools such as PowerFault-IDDQ are able to analize the results of multiple simulations (test benches) and optimize IDDQ test vector selection for highest fault coverage.

IDDQ test tools do not have access to actual IDDQ current values. With the exception of gross IDDQ testability violations, actual current values are not necessary to determine which functional test vectors qualify for IDDQ testing. IDDQ tools sort functional test vectors by “Qualified” or “Not qualified” status. This is done by monitoring the logic simulation to identify: 1) unitialized nodes, 2) floating nodes, 3) nodes values in contention and, 4) pullup/pulldown resistors active. Any of these conditions disqualify a functional test vector for IDDQ tests because quiescence cannot be guaranteed. IDDQ tools run from digital logic or fault simulators and do not have access to analog information.

7.1.1. IDDQ Test Strobe at the End of the Test Cycle

IDDQ test strobes will always be at the very end of the selected functional test vector. An IDDQ test is implemented by halting at the end of a functional test vector, before the next functional test vector is applied, wait for settling time, then measure IDDQ. The IDDQ strobe, therefore, will always be the last possible time before the next functional test cycle.

7.2. Test at High Voltage

Testing at high voltage limits increases the current through high impedance shorts.

7.3. Test at High Temperature

Testing at high temperatue stresses the DUT and is more effective at detecting faults.

---

8. Fault Models for IDDQ Test Tools

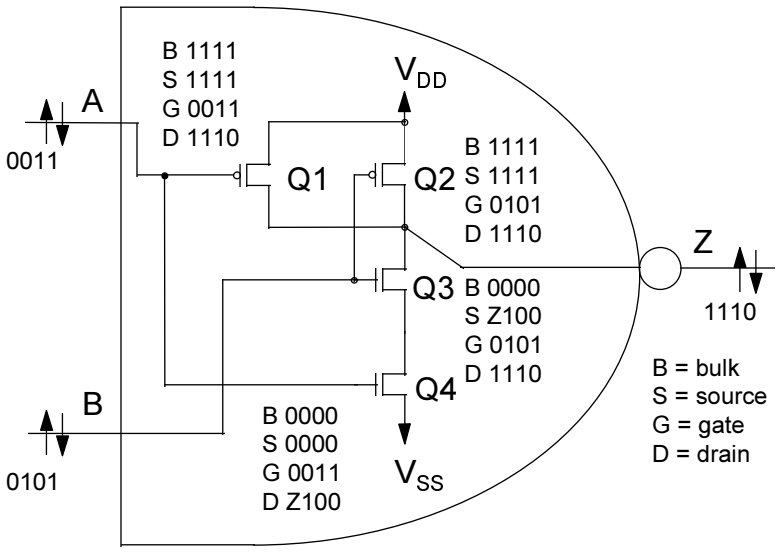


Figure 8 NAND Example of Model types

In the tables that follow:

D = Detected
U = Undetectable
N = QUIETEST detected faults that may not be detected by Cell Model

8.1. Toggle Model

The toggle fault model is the simplest and least comprehensive of all of the IDDQ defect models. For a two-input NAND gate, there are six faults, each input and the output stuck-at both '0' and '1'. The rule for counting a fault detected is that the faulted node, when tested, must be in the state opposite its stuck value. 100% fault detection is reported with input vector sets {00,11}, {00,01,11}, {00,10,11), and {01,10,11}. Each circuit node is a fault observation/detection point for fault grading. The Toggle model requires only a logic simulator to grade IDDQ faults. Toggle testing is a good starting point before any effort is made to fault simulate a test bench. This model has the same faults as the Pseudo Stuck-at model, but differs in that it does not require that faults be propagated to gate outputs.

In	Out	fault					
A B	Z	AS0	AS1	BS0	BS1	ZS0	ZS1
0 0	1		D		D	D	
0 1	1		D	D		D	
1 0	1	D			D	D	
1 1	0	D		D			D

Table 2 Toggle Detection Map, 6 Faults

8.2. Pseudo Stuck-at Fault Model

The Pseudo Stuck-at defect model is a carry over from the functional test stuck-at model. For a two-input NAND gate, there are six faults, each input and the output stuck-at both '0' and '1'. The rule for counting an input fault detected is that, when tested, its node be in the state opposite its stuck-at value, and that the gate's output be the value opposite its value when no fault is present. Gate outputs are detected just by the output being at the state opposite the stuck-at value. 100% fault detection is achieved with the input vector set {10,01,11}. IDDQ tools that use this model require a fault simulator. Each gate output is a fault observation/detection point for fault grading. All cells must be modeled at the gate level, otherwise accuracy is adversely affected. Model accuracy for simple combinational logic cells is equivalent to the QUIETEST model (Table 4). Notice that both input stuck-at 1 faults are **NOT** detected with input vector 00 (different than toggle model).

In		Out	faults					
A	B	Z	AS0	AS1	BS0	BS1	ZS0	ZS1
0	0	1					D	
0	1	1		D			D	
1	0	1				D	D	
1	1	0	D		D			D

Table 3 Pseudo Stuck-at Detection Map, 6 Faults

8.3. QUIETEST Fault Model

The QUIETEST Model was introduced in a paper by W. Mao, R.K. Gulati, D. Goel, and M. Ciletti, entitled "QUIETEST: A quiescent current testing Methodology for detecting leakage faults" Int. Ckt. Comp. Aid. Des. (ICCAD), 1990. Several IDDQ test tools are modeled after this work.

A special fault model must be made for each cell type. Each node, at the transistor implementation level, is analyzed for each vector, of the set of possible input vector combinations, to determine which faults are tested. For a typical two-input NAND gate, there are twenty-four faults. The rule for counting a fault detected is that it exists in the fault look-up table entry (transistor level fault mapping table) that corresponds with the input vector for the cell. 100% fault detection is achieved with the input vector set {10,01,11}. Every transistor node is a fault observation/detection point for fault grading. QUIETEST requires only a logic simulator to grade IDDQ faults. Also, it requires mapping faults from the transistor implementation of each cell type in the DUT to a fault look-up table. The greatest disadvantage to this model is the extraordinary effort required to develop and maintain an accurate fault model library. Often, QUIETEST tools are used with libraries that have not been mapped for the specific library used by the DUT, a primary source of inaccuracy.

See Figure 8 for an example of mapping QUIETEST faults. Each transistor element is considered shorted gate-to-source (GS), gate-to-drain (GD), source-to-drain (SD), gate-to-bulk/well (GB), source-to-bulk/well (SB), and drain-to-bulk/well (DB). Each input combination is propagated through the transistor model of the gate, and when two nodes are at opposite states ('0' or '1'), the corresponding shorted node or nodes are marked as tested for that input vector.

faults

IN	O	Q1						Q2						Q3						Q4					
AB	Z	GS	GD	SD	GB	SB	DB	GS	GD	SD	GB	SB	DB	GS	GD	SD	GB	SB	DB	GS	GD	SD	GB	SB	DB
00	1	D	D		D	U		D	D		D	U			D				D					U	
01	1	D	D		D	U						U					D	N	D		D	N		U	D
10	1					U		D	D		D	U			D	N			D	D	D		D	U	
11	0		D	D		U	D		D	D		U	D	D	D		D			D	D		D	U	

Table 4 QUIETEST Detection Map, 24 Faults

8.4. Cell Boundary Fault Model

The Cell Boundary Model models input and output stuck-at faults, input-to-input bridges, and input-to-output bridges. For a two-input NAND gate, there are nine faults. They cover most of the faults covered by the QUIETEST model. The rule for counting a fault detected for the stuck-at faults is that the faulted node, when tested, must be in the state opposite its stuck value. For bridging faults, the involved nodes must be at opposite states. 100% fault detection is achieved with input vector sets {00,01,11}, {00,10,11} and {01,10,11}. Each circuit node is a fault observation/detection point for fault grading. The Cell Boundary Model requires only a logic simulator to grade IDDQ faults. It does not use special fault library models.

Compared to the QUIETEST model, the Cell Boundary is very slightly more lenient. Fortunately, the worst case possibility of uncovered QUIETEST faults, Q3-SD, Q3-SB, and Q4-SD not being detected is 33%, while all of the transistor interconnect faults are covered (Table 4). With tests limited to just a few IDDQ vectors, the chance of having three unique input vectors for any one gate is low.

Advantages of the cell model are: 1. No IDDQ library requirements (it is unnecessary to know the internal structure of a cell), 2. Ability to optimize test vector selection (more significant for high fault coverage with few vectors than the fault model type), and 3. Cell bridging faults have a high probability of correlating to real device defects.

IN	O	faults								
AB	Z	A0	A1	B0	B1	Z0	Z1	AB	ZA	ZB
00	1		D		D	D			D	D
01	1		D	D		D		D	D	
10	1	D			D	D		D		D
11	0	D		D			D		D	D

Table 5 Cell Model Fault Map, 9 Faults

9. Design Considerations for IDDQ Testability

Here are some common IDDQ testability problems with options to overcome some of the limitations.

9.1. Mixed Signal Designs

Provide separate power supply pins and rails for CMOS, analog, and bipolar sections of the design. This allows IDDQ to be measured on the CMOS section independently of the other sections. Analog and Bipolar circuits that cannot be isolated from CMOS can render the entire circuit IDDQ untestable.

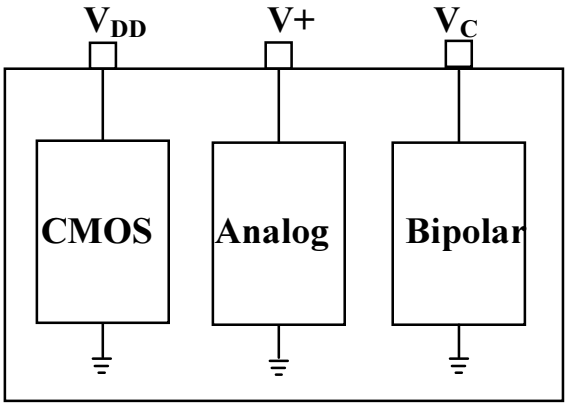


Figure 9

9.2. Separate Core Logic from I/O

To avoid leakage current from the tester's loads (see Figure 5) through the DUT's output pins, turn off the tester's loads, add buffers to DUT's output pads, or disconnect DUT's outputs via relays for IDDQ tests. Input pin leakage is insignificant. It is good practice to have separate VDD pins to supply core logic and I/O pads.

9.3. Avoid Pullups/Pulldowns

Whenever possible, use "keeper" latches instead of pullups/pulldows. When their use is unavoidable:

- Connect the pull resistor through a pass transistor that is controlled by a "test mode" signal.

- At primary input pins, try to have only pullups or pulldowns in the design. For non-complementary inputs implemented with only pullups, it may be possible to test ISSQ (the return side of the VDD supply) without loss of coverage. For pulldowns, test IDDQ. The current sink/source in both cases is from outside of the DUT, and by measuring the supply rail opposite the resistor source, the resistor current is bypassed. Where pullups or pulldowns cannot be effectively disabled, IDDQ can be measured only when the pulled net is in the same state as the pull resistor (i.e. for pullups, IDDQ can be measured only when the pulled net is '1').

9.4. Control Free-running Oscillators

Provide test mode access to disable free-running oscillators.

9.5. Avoid Contention

- If you must use internal tri-state buses, design a "test-mode" with which these buses can be held in a driven configuration for IDDQ. Do not design a case where multiple, separate internal tri-state busses are mutually exclusive.
- Do not allow nodes to float due to all drivers on a tri-stated bus being turned off. Do use bus repeaters ("keeper" latches) to hold values when the bus is not driven.
- Do not allow internal drive fights due to multiple enabled drivers on one bus. Do multiplex the driver enable ports to insure that only one driver is active at any given time.
- Do not allow degraded voltages, such as the result of using an n-transistor serving as a pass gate.

9.6. RAM

Provide test mode to disable RAM speed-up currents or to put RAM in standby mode.

9.7. Modeling Issues

- Model pullups structurally. Do not use behavioral assignments.
- Identify "leaky" conditions for behavioral models.

9.8. Deep Sub-micron

9.8.1. Scaling

Reduction of power supply voltages causes reduced threshold voltages. This results in increased off current. The overall scaling problems cause problems both in the ability to test for IDDQ and in product reliability. Technology that enhances product reliability also tends to enhance IDDQ testability (such as SOI).

DUTSs less than .5 micron and well above 1M transistors are successfully IDDQ tested.

9.8.2. Density

For designs above 1M transistors the sum of each transistor's leakage could approach the value of the IDDQ test's threshold value. Partitioning the design into smaller sections, each with separate power pins, can help to avoid the loss of IDDQ sensitivity.

10. Summary

10.1. IDDQ is a Cost-Effective Method for Testing CMOS Circuits

This methodology can use Design Verification simulation results to generate high quality test sets. These tests can easily be developed in time to test prototype devices. As illustrated in Figure 4, IDDQ tests are capable of detecting more defects than any other single test methodology. Ideally, all of the test methods discussed will be implemented, but since their actual implementation may be a serial process, overall quality is enhanced by starting with IDDQ.

10.1.1. Few Vectors

Most ASIC vendors will test 5-20 IDDQ test vectors for their customers. The long test time associated with IDDQ tests tends to restrict the number of test vectors allowed. Fortunately, IDDQ tests are capable of achieving high coverage with few vectors, as illustrated in Figure 3.

10.1.2. All Vectors

Some critical applications require "zero-defect" test methodology. For these applications, where test time is not an issue, all IDDQ qualified functional test vectors are tested for IDDQ.

10.1.3. Two IDDQ Vector Sets

Some test programs consist of two sets of IDDQ tests. One has only a few vectors and is used for production tests. The second test uses many or all vectors and is used for failure analysis and fault isolation.

10.1.4. Supplemental Vectors

IDDQ test vector sets are often supplemented by tests generated specifically to detect device failures that have escaped all manufacturing test. After diagnostic analysis that isolates and locates these failures, test engineers create supplemental test vectors.

Systems Science's PowerFault-IDDQ™

PowerFault-IDDQ™ is a "push-button" IDDQ solution for Verilog designs. It finds near-optimal IDDQ vectors and generates detailed fault coverage information.

PowerFault-IDDQ models faults at the cell level, allowing it to work with Verilog libraries, netlists and testbenches without modification. It is able to both read and write Verifault or Zycad seed fault lists. It uses the same Verilog simulator as is used for sign-off verification and the same sign-off Verilog ASIC libraries.

KEY BENEFITS

- Increase fault coverage and quality
- Fast and accurate
- Easy to use - simply run the Verilog simulator
- Handles any Verilog circuit
- No new libraries - uses ASIC vendor's Verilog library

KEY FEATURES

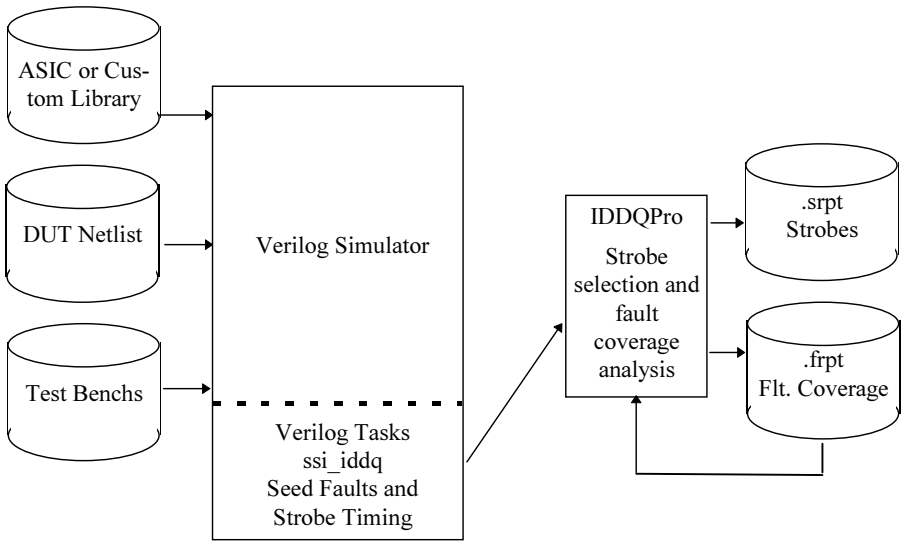
- Automatic seeding of "stuck-at" and "bridging" faults
- Seeds post-layout "bridging" faults
- Works with "Scan" and "Non-Scan" designs
- New algorithms find near-optimal set of IDDQ vectors quickly
- Optimal test vector selection over single or multiple test benches
- Can increase coverage beyond that of other IDDQ tools
- Links with Verilog-XL and many clones
- Generates detailed coverage reports
- Can generate hierarchical coverage reports
- Interactive or batch analysis mode
- Robust Verifault and Zycad interfaces

Overview of PowerFault-IDDQ

PowerFault-IDDQ is linked with Verilog-XL, or compatible clones, through the PLI. Since it does not require that another simulator be introduced to the design environment, it eliminates the need to compare results with another simulator. This scheme assures accuracy in simulation and assures the highest probability that the IDDQ tool can properly identify quiescent test vectors.

PowerFault-IDDQ is easy to use. It is executed in two stages:

1. Insert □□□□□□□□□□ systems tasks into a top level Verilog module. Run your Verilog simulation with the normal Verilog testbench.
2. After simulation, run the IDDQ profiler, IDDQPro. It can be executed interactively or in batch mode. The profiler analyzes output produced by the □□□□□□□□□ tasks and selects near optimal IDDQ test vectors. The profiler also produces hierarchical fault coverage reports.



PowerFault-IDDQ Data Flow

PowerFault-IDDQ is Easy to Use

Just add a few systerms tasks:

□□□□□□□□□□□□

□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

□□□

□□

□□□□□□□□□

Rich set of options include:

- Cell bridging faults
- Layout bridging faults
- Selective hierarchical fault seeding
- Use Verilog or Zycad fault seed lists

PowerFault-IDDQ is Cost-effective:

	Others	PowerFault-IDDQ
Install New Simulator	$ day(s)	No
Library Development	$$$ Months	No
Training	$$ Week(s)	Self-Teach
Useable Results	$$$ Months	First Day

VERIFICATION & TEST SOLUTIONS

Are You Sure Your Design Is Verified For Tape Out?

ATTN: All Verilog, ASIC and Systems Designers, Foundries, and IP Developers...
Verify Correctness. Develop Testbenches Quickly. Catch Design Problems Early. Cycle Based and Regular Timing. Avoid Verilog Re-Compiles Between Testbenches. Object Oriented Options. Mix and Match Verilog, Vera, C and C++.

VERA

Can You Afford Field Failures?

ATTN: All ASIC Designers, Test Engineers and Foundries...
Use Standard Verilog Libraries and Standard Verilog Simulators. "Bridging" and "Stuck-At"Faults. Easy to Use. Unmodified Netlists and Libraries. Optimal Test Vectors. Increase Quality. Verifault and Zycad Compatible.

POWERFAULT-IDDQ

Do You Have Too Many Waveform Displays?

ATTN: All EDA and Test Vendors and Internal CAD Groups...
Displays Digital, Mixed-Signal and Analog Data. Most Powerful Waveform Tool in EDA. OEM'd and Supported by Industry Leaders such as Cadence, Mentor Graphics, Epic, Zycad, Precedence, Meta Software and Others. Standardize Using SimWave with ISDB.

SIMWAVE

Does Your Simulator Crawl When You Dump Results?

ATTN: All EDA and Test Vendors and Internal CAD Groups...
Fastest and Most Compact Database. Support for Verilog, VHDL, Analog and More--Open API. Simple Integration. Platform-Independent. Interactive and Batch Operation. The Industry Standard Simulation and Test Database.

ISDB

SYSTEMS SCIENCE

Systems Science Inc.

PROFILE

Systems Science Inc. (SSI) develops and markets advanced Electronic Design Automation (EDA) software tools. Today, SSI's tools are in use at more than 6000 seats in the semiconductor, computer, and electronics industries worldwide. SSI was founded in 1986.

FOCUS

SSI's tools are designed to operate in conjunction with Verilog and VHDL simulators, with analog tools, such as Spice and PowerMill, and with hardware accelerators. They run on UNIX workstations, including SPARC, HP700, RS6000, DEC Alpha, SGI, and PowerPC.

CLIENTS

A partial list of major clients:

		Intel	National	Sanyo	Toshiba
AT&T	Ericsson	LSI	Nissan	SGS Thomson	Unysis
AMD	Fujitsu	Matshushita	Nortel	Sharp	VLSI
Alcatel Boeing	Goldstar	Mitsubishi	OKI	Siemens	Xerox
Brooktree	Hitachi	Motorola	Olumpus	SMOS	Yamaha
Casio	HP	NASA	Philips	Sony	
DOD	Hyundai	NCR	Ricoh	SUN	
Delco	IBM	NEC	Samsung	TI	

OEM/PARTNERS

Cadence Design Systems Inc. | Veda Design Automation Ltd.
Zycad Corporation | Epic Design Technology, Inc. | Pendulum Design Inc.
Meta-Software Inc. | Anagram Inc. | Simplex Solutions Inc.

TECHNOLOGY PARTNERS

Sun Microsystems Inc. | NEC Electronics, Inc. | Hitachi America Ltd.

DISTRIBUTORS

Japan - Marubeni Hytech Corp. | Europe - Veda Design Automation Ltd.
Taiwan - Avant Technology Inc. | Singapore - SPSDA Ltd.
Korea - Stealth Systems Corp. Ltd. | Israel - Wizard Ltd. | U.S. Midwest - Silicon Software Solutions