Iddq Test Pattern Generation for Scan Chain Latches and Flip-Flops

Samy R. Makar and Edward J. McCluskey*
Center for Reliable Computing
Stanford University
Stanford, CA 94305

Abstract

A new approach, using Iddq, for testing the bistable elements (latches and flip-flops) in scan chain circuits is presented. In this approach, we generate test patterns that apply a checking experiment to each bistable element in the circuit while checking their response. Such tests guarantee the detection of all detectable combinational defects inside the bistable elements. We show that this approach is more effective than test generation using the popular pseudo stuck-at fault model. Our algorithm was implemented by modifying an existing stuck-at combinational test pattern generator. The number of test patterns generated by the new program is comparable to the number of traditional stuck-at patterns. This shows that our approach is practical for large circuits.

1. Introduction

Traditional test approaches use scan as a method to access inputs and outputs of internal combinational logic, so that patterns can be applied directly to the inputs of combinational logic, and the responses can be captured from the outputs of this logic. Stuck-at patterns are often used. The patterns are usually generated by combinational ATPG tools if full scan is used, or sequential ATPG tools if partial scan is used. Even though high stuck-at coverage can be easily achieved using these methods, the final quality is sometimes below desired levels. Iddq is used as an economic technique to detect defects missed by the stuck-at test. However, even the Iddq test often targets faults in the combinational logic, and the scan chain is used simply as a means of transferring the patterns to the input of the combinational logic. Faults inside the bistable elements are often ignored.

Recent work [1] and [2] have stressed the importance of using Iddq for testing latches and flip-flops. Both papers show that boolean testing alone of flip-flops is not sufficient for detecting all the internal defects. Earlier works, [3], [4] and [5], showed that bistable internal faults can be missed by traditional scan tests.

In [2] the Iddq testability of different flip-flop implementations was analyzed. Not much discussion was given to the test applied. In fact, the test simply loads 1, loads 0, and exercises the asynchronous set and reset signals once. This test was able to detect the bridging faults discussed in that paper. However, it can miss other important faults, such as stuck-opens and open-interconnects. Analysis of circuits and faults at the transistor level for Iddq testability can easily result in different patterns for different implementations of the same flip-flop type. This makes test generation for a circuit using a mixture of different flip-flop implementations very difficult as it would involve transistor level analysis. Designers may use different implementations of flip-flops to satisfy design performance and area requirements. What is really needed is a universal Iddq test for all implementations of a particular flip-flop type, and a method for getting this pattern to the flip-flop under test.

Iddq test generation and fault simulation is often based on the pseudo stuck-at model [6]. In this model, stuck-at faults on the inputs of logic gates are considered detected if the fault is sensitized, and its effect is propagated through the gate. The fault effect need not propagate any further. This model works well for fully complementary CMOS gates, because there is a clear relation between the pseudo stuck-at fault and transistor faults within the gates. However, no such relation exists for transmission gates and sequential cells.

This paper introduces a new approach for testing the bistable elements in scan chain circuits. Our test is based on checking experiments. A *checking experiment* is a sequence of inputs that, when applied to a circuit, gives different outputs than any other circuit with the same inputs, outputs, and same number of or fewer states [7]. The main advantage of a checking experiment approach over other methods (such as stuck-at ATPG) is that it is independent of the fault model, and will detect any defect that does not increase the number of states in the circuit. The main problem with checking experiments is that the number of test patterns can be very large, making it impractical for large circuits. However, we use a checking experiment only for the bistable elements. We show that such tests are comparable in size to stuck-at tests, indicating that they are practical for large circuits.

We introduced our checking experiment test generation technique in [8]. In that paper we focused mainly on generating patterns that would test the scan chain better than the traditional stuck-at test approach. In this paper, we focus on test generation for Iddq faults.

The rest of this paper is divided as follows. In Section 2, we describe the basis of our algorithm, and in Section 3, we describe an implementation of our algorithm. In Section 4, we present the fault simulation results for a single MD flip-flop. The simulations include test patterns for stuck-at faults and checking experiment based test patterns. The results indicate that there are many faults

---

* Samy is currently with Cirrus Logic Inc.

missed by the stuck-at test, that are detected by our test. Also, we see that many of the faults require Iddq measurement to be detected. In the same section, we present test generation results for all the ISCAS-89 benchmark circuits [9]. Section 5 concludes the paper.

2. Checking Experiments for Bistable Elements

Flip-flops can have many flow tables, but there is only one primitive flow table (up to isomorphism) for each flip-flop type [10]. Therefore we use primitive flow tables in our analysis. In a *primitive flow table*, each row contains only one stable state. A checking experiment for a primitive flow table will detect all defects that do not increase the number of states in any column.

Given the primitive flow table of a bistable element, we can easily derive a checking experiment for the bistable element. However, if a bistable element is embedded inside a circuit, we need to find a way to get the checking experiment from primary inputs to the bistable element, and to observe the output of the bistable element under test from the primary output. This may be very difficult or impossible to do. Also, there are many possible checking experiments. Fig. 2-1 shows a small circuit with three flip-flops. We will show how to generate a checking experiment for the shaded MD flip-flop (F_3).

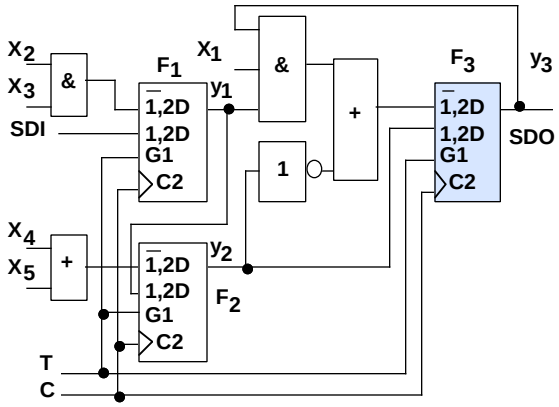

Figure 2-1 Circuit Under Test.

Instead of trying to apply a complete checking experiment directly from primary inputs, we use a divide-and-conquer approach. Every checking experiment must identify all the stable and unstable states in its primitive flow table. For each of these states a sequence of inputs must be applied to the bistable element under test.

For each bistable element in the circuit, several sub-sequences are needed. These sub-sequences can be derived from the primitive flow table. If all the sub-sequences for a bistable element are applied to the bistable element under test, then any combinational defect is guaranteed to be activated [11]. Therefore, a test that applies all the sub-sequences to the bistable element under test detects any combinational defect that is Iddq testable. So, detecting the pseudo-faults of the bistable element under test detects all the Iddq testable defects in the bistable element.

An example of an input sequence for an MD flip-flop is shown in Fig. 2-2. We can think of the pseudo-fault corresponding to this sequence, as a fault requiring the sequence of values shown in Fig. 2-2. We can use the scan chain to supply a test pattern that would set the inputs and output of the flip-flop under test to the initial values (d=0, s=1, and q=1).

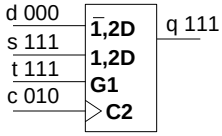

Figure 2-2 A Required Input Sequence for an MD Flip-Flop.

The method for finding a test pattern that would set d=0, s=1, and q=1 is similar to combinational ATPG. In our example, the test pattern $X_1 = 0$ and $y_2 = y_3 = 1$ would set d, s and q to the desired values (see Fig. 2-3). After applying C = 010, d and s will depend on the new values in F_1 and F_2. Thus, the test pattern we use must preserve s = 1 and d = 0 after the clock pulse. In our example, this can be done with the test pattern $X_1 = 0$ and $y_1 = y_2 = y_3 = 1$ (see Fig. 2-4). The difference between this and the earlier test pattern is that y_1 is set to 1, so that after the clock pulse y_2 is still 1.

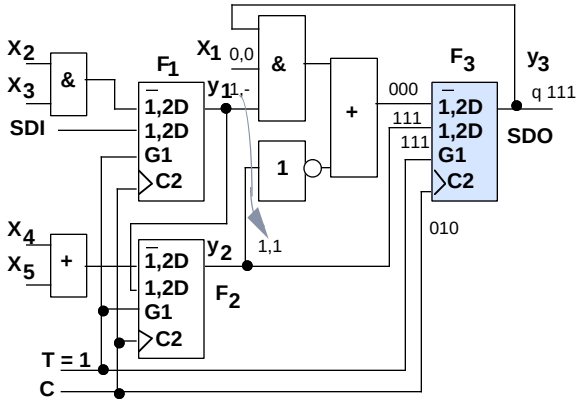

Figure 2-3 Circuit Under Test With Test Pattern.

This analysis is similar to sequential ATPG because more than one time frame is considered, i.e., we consider the values on the flip-flop before and after the clock pulse. Since we had only one pulse on C, we have to deal with only two time frames. This makes the problem much easier than general sequential ATPG. In the above example, we had T = 1, and applying a pulse on C caused the scan chain to shift once. Thus the operation of generating a test pattern for such a sequence is called a shift operation. Other required sequences need other types of operations. These elementary operations are summarized in Table 2-1. The first two elementary operations are used with sequences for which there is no pulse on the clock. The third (the one we showed in the example) and fourth are used with sequences in which there is a pulse on the clock.

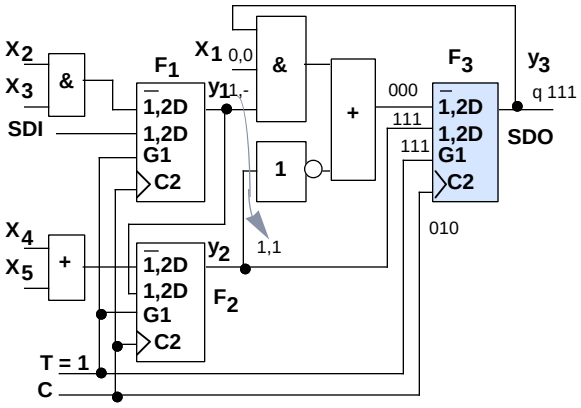

Figure 2-4 Circuit Under Test With Test Pattern.

Table 2-1 Elementary Operations.

Op	Description
Single Cycle	Determine bit values of a test pattern that would set lines in the circuit to desired values.
Single Cycle Change	Determine bit values of a test pattern that would set lines in the circuit to desired values, and by changing values only on the primary inputs would change the value of a line in the circuit.
Shift Operation	Determine bit values of a test pattern that would set lines in the circuit to desired values, and after the scan shifts by one, would again set some lines in the circuit to desired values. The values on the lines need not be the same for both cycles
Normal Operation	Determine bit values of a test pattern that would set lines in the circuit to desired values, and after a normal cycle (bistable element input selected from combinational logic), would again set some lines in the circuit to desired values. The values on the lines need not be the same for both cycles.

Elementary operations can be used to generate test patterns for all the required input sequences [11]. Therefore, we can define an algorithm for generating test patterns for the bistable elements using the algorithm in Fig. 2-5. In this algorithm, we use elementary operations to find a test pattern for each required sequence of each flip-flop in the circuit. The test patterns are placed in pattern tables that are compacted using standard test pattern compaction techniques. There are four common scan chain architectures [10]. The architectures use different bistable elements for scan cells. Different bistable element types have different required sequences, and thus even though their algorithms have the structure in Fig. 2-5, each will have a different implementation.

```
for each bistable element {
  for each required sequence {
    Apply Appropriate Elementary Operation
    Add Test Pattern To Appropriate Table
  }
}
Compact Tables
Print Tables
```

Figure 2-5 Algorithm for ATPG for Bistable Elements.

3. Implementation

We implemented our algorithm by modifying an existing stuck-at ATPG program in SIS [12]. This was done by first creating functions for the elementary operations and then using these functions to write procedures for the four different bistable element types used in the scan chain architectures.

As with most ATPG programs, this program reads a gate level description of the circuit. However, unlike most ATPG programs, the output is not simply a file with test patterns, but rather a set of files with test patterns. The number of test pattern files depends on the scan architecture used. Each file of patterns corresponds to a different type of sequence that has different timing on the clock and control inputs. Details can be found in [11].

4. ATPG Results

The effectiveness of a test can be measured by the number of defects it can detect. Even though the stuck-at models are often used for fault simulation, we use the more accurate (for CMOS circuits) CrossCheck fault models, [13] and [14], for our simulation. The fault models comprise shorted interconnects (STI), open interconnects (OPI), short-to-power (STP), short-to-ground (STG), transistor stuck-on (SON), and transistor stuck-open (SOP). In the simulations, faults are injected by modifying a copy of the circuit description. The faulty circuits were simulated using HSpice [15].

In CMOS, there are some faults whose presence does not change the functionality of the host circuit. Some of these cannot be detected (and thus are untestable or redundant). Others that cannot be detected by a Boolean voltage test (since the circuit functionality is correct) can, nevertheless, be discovered by a current test or a delay test [16]. The simulations reported here record whether tests caused excessive supply current (Iddq) or incorrect outputs. The current limit for Iddq testing is often determined experimentally, by plotting the values of many good and bad dies, and selecting an appropriate threshold that would detect as many faulty circuits as possible without discarding many good ones [17] and [18]. For our simulations, the current limit is determined by plotting the maximum observed current for each fault, and selecting an appropriate threshold.

In Section 4.1, we present simulation results for an MD flip-flop, comparing traditional tests with checking experiment based tests. In Section 4.2, we compare ATPG results of a pure boolean approach (as described in [8]) with the Iddq approach described here.

4.1 MD Flip-Flop Fault Simulation

Three different tests for the MD flip-flop were simulated using HSpice. The first test, a traditional test, is based on scanning in and out the 01100 test pattern, and test patterns that would detect stuck-at 0 and stuck-at 1 faults on the D input of the flip-flop. The second test is a pin fault test set, which targets stuck-at faults on the input and output of the MD flip-flop. The third test is a checking experiment for the MD flip-flop in a scan chain [11]. The flip-flop

implementation used for the simulation is shown in Fig. 4.1-1. This implementation is selected because it is a commonly used structure.

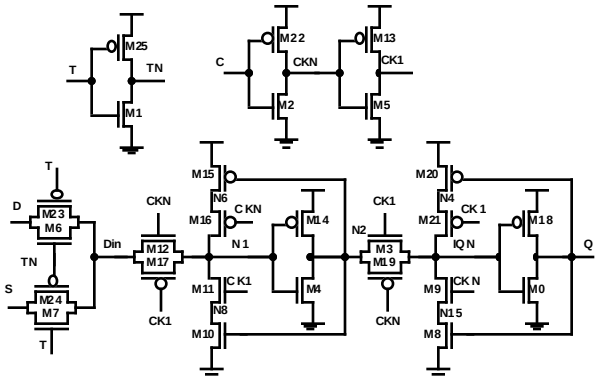

Figure 4.1-1 MD Flip-Flop Implementation Used in Simulation.

Table 4.1-1 Number of Faults Detected in MD Flip-Flop (Total Faults = 256).

	Boolean and IDDQ	Boolean Alone	IDDQ Alone
Traditional Test	212	167	155
Pin Fault Test	227	184	161*
Checking Exp.	237	207	182

* detection by pseudo stuck-at test.

The results of the simulations are shown in Table 4.1-1. From the table, there are 19 faults that were not detected by the checking experiment. These faults are shown graphically in Fig. 4.1-2. In this figure, white ovals indicate SON or OPI faults, black ovals indicate SON faults, and thick black lines indicate STI faults. All STP and STG faults are detected by all tests.

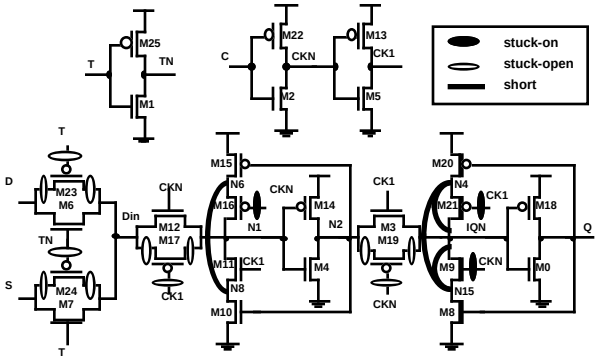

Figure 4.1-2 Faults Missed by Checking Experiment of MD Flip-Flop (19 of them).

The faults missed by the checking experiment fall into two groups. The first group of faults missed by the checking experiment is the SOP open faults on the transmission gates. These faults, though undetectable, could add a delay to the circuit, and will thus behave as delay faults. A test pattern that would detect a path delay fault through the input of the flip-flop may be able to detect these faults. The other group of faults missed by the checking experiment, the SON and STI faults, will turn the master or slave latch into a dynamic latch. Since a dynamic latch cannot guarantee holding its value for a very long time, then loading a value and waiting a long time may change the value in the flip-flop and the fault would be detected. Thus a very slow test (data retention test) is needed for these faults.

The traditional test and the pin fault tests miss many faults (about 5 %) detected by the checking experiment.

There were 21 Iddq faults detected by our checking experiment test that were missed by the pin fault test. Since the pin fault test detects all the stuck-at faults on the inputs of the flip-flop, it is an accurate representation of the pseudo stuck-at fault model. The 21 faults missed by the pseudo stuck-at fault based test are shown in Fig. 4.1-3.

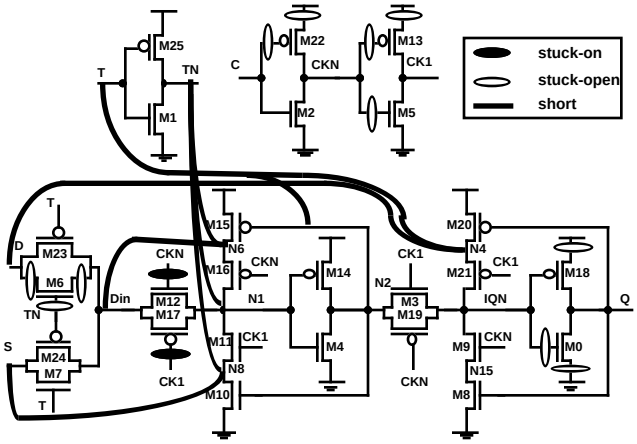

Figure 4.1-3 Faults Missed by Pseudo Stuck-at Test Detected by Checking Experiment (21 of them).

4.2 Circuits Using MD Flip-Flops

Of the four common scan architectures [10], only the MD flip-flop scan architecture requires sensitization of flip-flop outputs for a boolean test. Of course, for an Iddq test, no sensitization is required. In this section, we compare the tests generated for boolean (as described in [8]) and Iddq tests (as described in Section 3). The difference between the two tests is the sensitization of the flip-flop outputs. Iddq can, and should, still be measured for the test with sensitization, and the scan chain outputs can, and should, be checked for the test with no sensitization. The results are shown in Table 4.2-1. In this table each of the ATPG methods has three columns: the number of patterns (this is the number of patterns after test generation and compaction), the number of sub-sequences for which the program determined no pattern exists, and the number of sub-sequences that were aborted. It is easy to see that the program had an easier time with the no sensitization method. No sub-sequences were aborted, and there were always fewer inapplicable sequences. In general, the no sensitization method had fewer patterns. Exceptions occur in circuits with too many inapplicable or aborted sequences in the sensitization method.

One advantage of the sensitization method is that it can better test boolean output. This is important because, as

Table 4.1-1 indicates, there are some faults that can only be detected by a boolean test. This suggests a hybrid algorithm that would attempt to generate patterns with sensitization, but if it fails it would generate a pattern without sensitization. Results of this are shown in the last columns of Table 4.2-1.

Table 4.2-1 Number of Test Patterns for MD Flip-Flop Scan Architecture.

	With Sensitization			Without Sensitization			Hybrid Approach		
Circuit	P	I	A	P	I	A	P	I	A
S27	103	42	0	107	23	0	116	23	0
S298	302	117	0	274	115	0	305	115	0
S344	341	81	2	341	48	0	348	48	0
S349	313	81	2	314	48	0	317	48	0
S382	478	168	19	455	159	0	484	159	0
S386	232	80	0	244	77	0	236	77	0
S400	477	168	19	451	159	0	481	159	0
S444	416	241	16	390	227	0	423	227	0
S510	282	8	0	274	8	0	282	8	0
S526	497	179	0	464	163	0	503	163	0
S641	533	143	66	516	143	0	565	143	0
S713	511	143	89	518	143	0	549	143	0
S820	275	5	0	293	5	0	276	5	0
S832	274	5	0	292	5	0	275	5	0
S1196	570	17	124	487	10	0	599	10	0
S1423	1460	491	190	1324	461	0	1497	461	0
S1488	319	13	0	346	13	0	319	13	0
S1494	318	13	0	346	13	0	318	13	0
S5378	2294	1253	144	1692	1242	0	2341	1242	0

P = Number of Patterns, I = Number of Inapplicable Sequences, A = Number of Aborted Sequences.

5. Conclusions

We presented a new approach for generating Iddq tests for bistable elements. Some of the defects in the bistable elements can only be detected with Iddq. Our new approach is based on checking experiments for the bistable elements. Checking experiments are used because they guarantee the detection of all faults that do not increase the number of states. Since a checking experiment makes no assumption about the circuit implementation, it is implementation independent. This is especially useful since designers often use different implementations of bistable elements to optimize their circuits for area and performance.

Our test was compared with the traditional test by performing fault simulation of some of the bistable elements. The results clearly indicate that there are faults that traditional tests miss that are detected by our new test. We also showed that we detect many defects missed by the popular pseudo stuck-at model. In conclusion, tests based on checking experiments for latches and flip-flops are a thorough economic technique for testing the bistable elements of digital circuits. The checking experiment test provides excellent stimulus for Iddq testing.

Acknowledgment

The authors would like to thank Jonathan Chang, and Philip Shirvani for their valuable comments. This work was supported in part by the Ballistic Missile Defense Organization, Innovative Science and Technology (BMDO/IST) Directorate and administered through the Department of the Navy, Office of Naval Research under Grant No. N00014-92-J-1782, in part by the Advanced Research Projects Agency under Contract No. DABT63-94-C-0045, and in part by the National Science Foundation under Grant No. MIP-9107760. It was also funded in part by Cirrus Logic.

References

[1] Makar, S.R. , and E. J. McCluskey, "Some Faults Really Need an Iddq Test," *Digest of Papers, 1996 IEEE Int. Workshop on Iddq Testing*, pp. 102-103, 1996.

[2] Yamazaki, H., and Y. Miura, "Iddq Testability of Flip-flop Structures," *Digest of Papers, 1996 IEEE Int. Workshop on Iddq Testing*, pp. 29-33, 1996.

[3] Reddy, M.K. and S.M. Reddy, "Detecting FET Stuck-Open Faults in CMOS Latches and Flip-Flops," *IEEE Design and Test of Computers*, Vol. 3, No. 5, pp. 17-26, October, 1986.

[4] Lee, K.J. and M.A. Breuer, "A Universal Test Sequence for CMOS Scan Registers," *CICC*, pp. 28.5.1-28.5.4, 1990.

[5] Al-Assadi, W.K., "Faulty Behavior of Storage Elements and Its Effects on Sequential Circuits," *IEEE Transactions on VLSI*, Vol. 1, No. 4, December, 1993.

[6] Maxwell, P., et. al., "The Effectiveness of Iddq, Functional and Scan Tests: How Many Fault Coverages Do We Need?" *Proc. ITC*, pp. 168-177, 1992.

[7] Hennie, F.C., "Fault Detecting Experiments for Sequential Circuits," *Proc. of the Fifth Annual Switching Theory and Logical Design Symposium*, S-164, Princeton, New Jersey, pp. 95-110, 1964.

[8] Makar, S.R., and E.J. McCluskey, "ATPG for Scan Chain Latches and Flip-Flops," *Proc. VTS*, pp. 364-369, 1997.

[9] Brglez, F., D. Bryan and K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuit," *IEEE ISCAS Proc.*, pp. 1929-1934, 1989.

[10] McCluskey, E.J., *Logic Design Principles*, Prentice-Hall, New Jersey, 1986.

[11] Makar, S.R., and E.J. McCluskey, "Checking Experiments for Scan Chain Latches and Flip-Flops," *CRC Technical Report 96-5*, August, 1996.

[12] Sentovich, E.M., et. al., "SIS A system for Sequential Circuit Synthesis," *Electronics Research Lab* Memorandum, No. UCB/ERL M92/41, 1992.

[13] Sucar, H., "High Performance Test Generation for Accurate Defect Models in CMOS Gate Array Technology," *ICCAD*, pp. 166-169, 1989.

[14] Chandra, S., et. al., "CrossCheck: An Innovative Testability Solution," *IEEE Design and Test of Computers*, Vol. 10, No. 2, pp. 56-67, June, 1993.

[15] Kielkowski, R., *Inside SPICE Overcoming the Obstacles of Circuit Simulation*, McGraw Hill, USA, 1994.

[16] Ma, S., P. Franco and E.J. McCluskey, "An Experimental Chip to Evaluated Test Techniques, Experimental Results," *Proc. ITC*, pp. 663-672, 1995.

[17] Hawkins, CF., "Quiescent Power Supply Current Measurement for CMOS IC Defect Detection," *IEEE Trans. on Industrial Electronics*, pp. 211-218, May, 1989.

[18] Perry, R., "Iddq Testing in CMOS Digital ASICs - Putting It All Together," *Proc. ITC*, pp. 151-157, 1992.