Center for Reliable Computing

TECHNICAL NOTE

Preprint

Makar, S.R., and E.J. McCluskey, "ATPG For Scan Chain Latches and Flip-Flops."

96-3 (CSL TN # 96-405) November 1996	**Center for Reliable Computing** Gates 2A Computer Systems Laboratory Departments of Electrical Engineering and Computer Science Stanford University Stanford, California 94305-9020

Abstract:

This Technical Note contains a preprint of a paper submitted to the 15th IEEE VLSI Test Symposium to be held on April 27-30, 1997 at Monterey, CA.

Funding:

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.

ATPG For Scan Chain Latches and Flip-Flops

Samy Makar and Edward McCluskey
Center for Reliable Computing
Computer System Lab
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, CA 94305

Tel: (415) 723-1258
Fax: (415) 725-7398
e-mail: samy@shasta.stanford.edu, ejm@shasta.stanford.edu

Suggested Topics: ATPG, Scan Chain, Functional Faults.

Abstract

A new approach for testing the bistable elements (latches and flip-flops) in scan chain circuits is presented. This approach guarantees the detection of all detectable combinational defects inside the bistable elements.

ATPG For Scan Chain Latches and Flip-Flops

Samy Makar and Edward McCluskey

Abstract

A new approach 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 in the circuit and observe their response. Such tests guarantee the detection of all detectable combinational defects inside the bistable elements. The algorithm is 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 this approach is practical for large circuits.

Introduction

Scan was introduced to overcome the difficulties of sequential test generation [Williams 73], [Eichelberger 77]. The basic idea of scan is to allow easy access to the flip-flops in the design so that test patterns can be applied directly to the inputs of the internal combinational logic, and the outputs of the internal combinational logic can be "captured" by the bistable elements and scanned to the primary output for comparison with expected values. This makes it possible to use combinational circuit test generation algorithms on sequential circuits.

A difficulty with scan based methods is that they do not address faults within the bistable elements. [Reddy 86], [Lee 90] and [Al-Assadi 93] showed that tests for stuck-at faults at flip-flop inputs and outputs miss many internal faults. Simulation results presented in this paper confirm these results. The three bit binary counter shown in Fig. 1-1 was used in our simulations. We found 37 faults that were not detected by a 100% stuck-at test, but affect the normal operation of the counter. Fig. 1-2 shows some of these faults. All these faults are detected by our test.

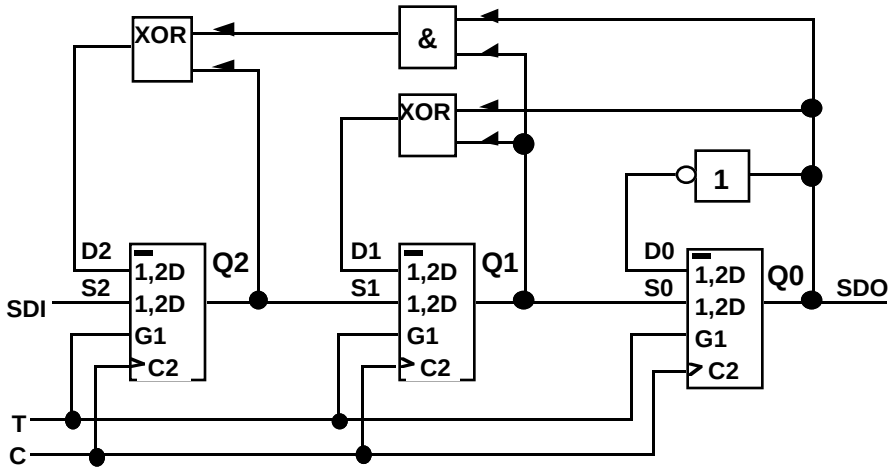


Figure 1-1 Three Bit Binary Counter.

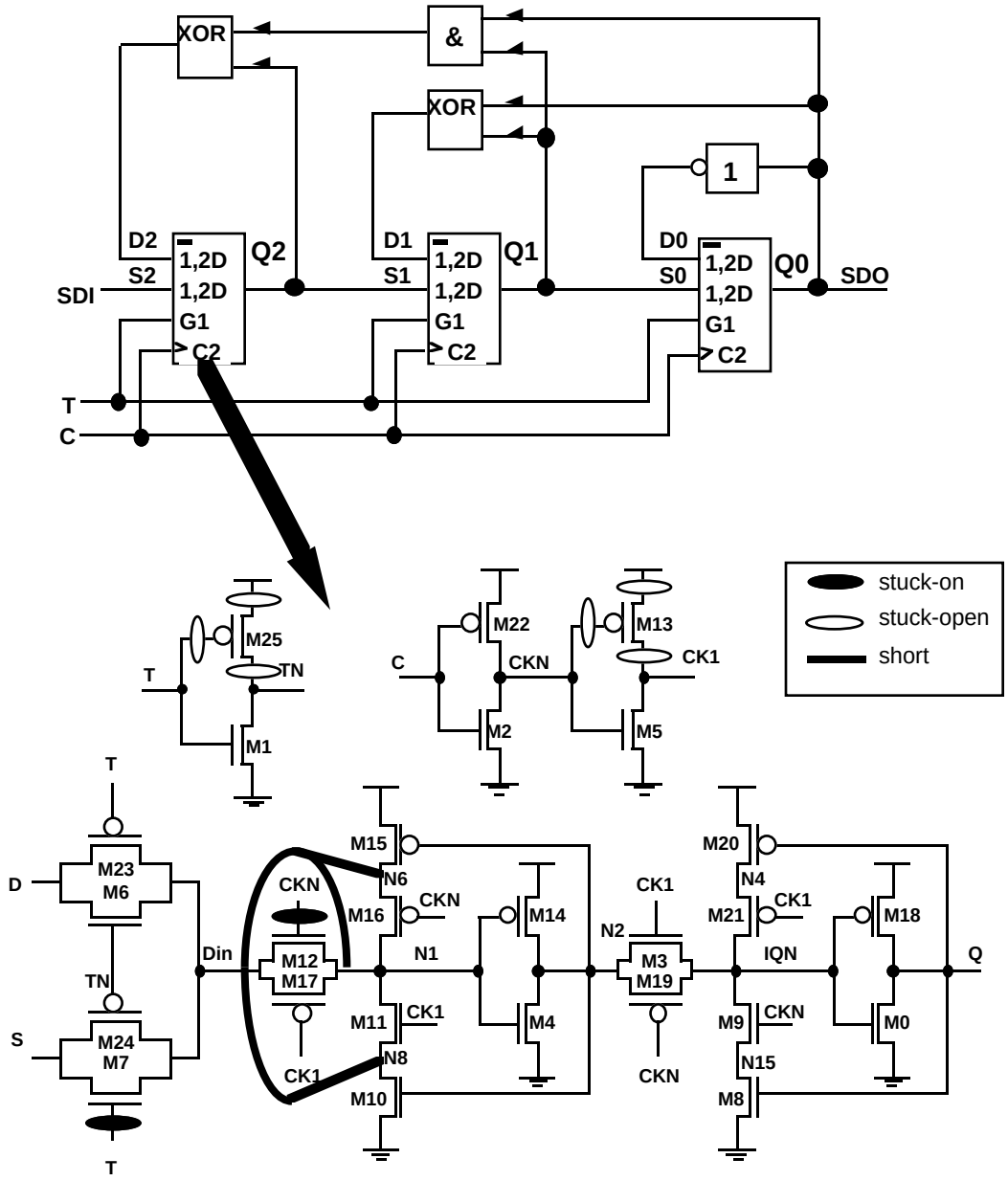


Figure 1-2 Faults in First Flip-Flop of Binary Counter Missed by 100% Stuck-At Test That Affect Normal Operation.

Our 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 input, outputs, and same number of or fewer states [Hennie 64]. 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 for even small circuits, 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 presented some early results of this work in [Makar 95]. That paper described a technique for finding tests for scan chain latches, but cannot be used for flip-flops. The technique described here applies to both flip-flops and latches. Another important difference is that here we implemented our algorithm and generated tests for all the ISCAS-89 circuits [Brglez 89]. The test lengths of our tests were comparable with stuck-at test lengths.

The rest of this paper is divided as follows. In Section 2, we present some important definitions, including the four common scan architectures [McCluskey 86]. In Section 3, we describe the basis our algorithm, and in Section 4, we describe an implementation of our algorithm. In Section 5, we present fault simulation results of 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 missed by the stuck-at test. In the same section, we present simulation results of a three bit binary counter with faults in the MD flip-flops. Test patterns used here are a traditional test and a test generated by the program described in Section 4. Again, many faults are missed by the traditional test that are detected by our test. We also generate test patterns for all the ISCAS-89 benchmark circuits [Brglez 89]. The number of test patterns is compared with traditional stuck-at patterns. Results indicate that the number of test patterns from our program increase at the same rate as stuck-at patterns. This implies that they are practical for large circuits.

2. Definitions

There are several architectures for scan designs: multiplexed-data latch based architecture (Fig. 2-1), level sensitive scan design architecture (Fig. 2-2), multiplexed flip-flop based architecture (Fig. 2-3), and two-port flip-flop based architecture (Fig. 2-4) [McCluskey 86]. The bistable elements used in these architectures are shown in Table 2-1. There are two modes of operation in any scan design: shift mode and normal mode. In *shift mode*, the scan chain is

Table 2-1 Definitions of Bistable Elements used in Scan Architectures.

Bistable Element	Definition
D-Latch	A sequential element, in which the data input is propagated to the output when the clock is active, otherwise it holds the stored value.
Multiplexed-Data latch (*MD-latch*)	A D-latch with multiplexed data inputs
Two-Port latch (TP-latch)	A D-latch two control inputs with the data source determined by the active control input.
Multiplexed-Data flip-flop (MD flip-flop)	A D flip-flop with multiplexed data inputs;
Two-Port flip-flop (TP flip-flop)	A D flip-flop two control inputs with the data source determined by the transitioning control input

configured as a shift register to scan data in and out of the bistable elements. In *normal mode*, the bistable elements are configured to get their inputs from the combinational logic and the circuit performs normal operation. In the MD-latch and MD flip-flop architectures, the shift mode is activated by setting T = 1, and normal mode is activated by setting T = 0. In the other two architectures, TCK (and CK_2) is used as the clock for shift mode, and CK_1 (and CK_2) is used for normal mode.

The combinational logic is tested by scanning in a test pattern with the scan chain in shift mode. Values at the primary input are then applied, and values at the primary output are compared with expected values. The scan chain is switched to normal mode for one cycle to capture the output of the combinational logic in the bistable elements. Then the scan chain is switched back to shift mode, and the contents of the bistable elements are scanned out and compared with expected values at SDO. The next test pattern is scanned in while the bistable element contents are being scanned out. The bistable elements themselves are tested by keeping the scan chain in shift mode

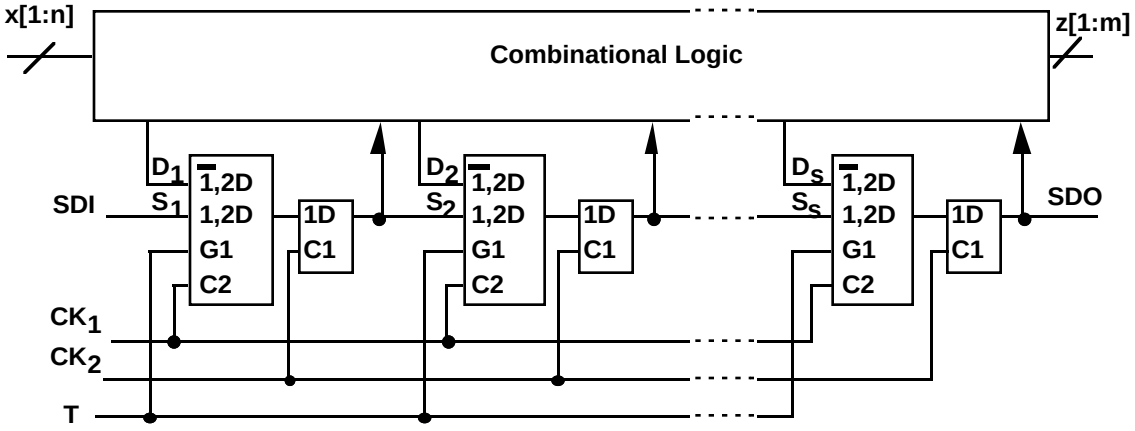


Figure 2-1 MD-Latch Based Scan Architecture.

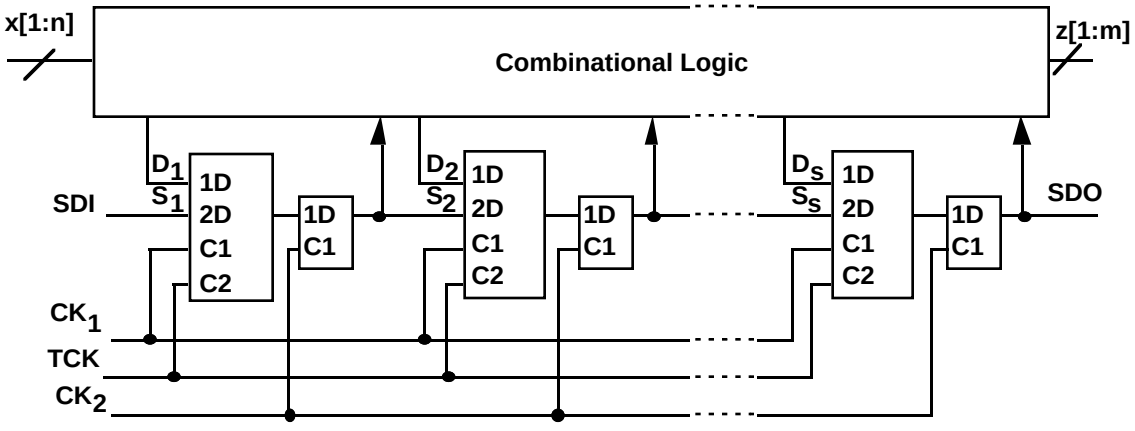


Figure 2-2 Level Sensitive Scan Design (LSSD) Architecture.

and shifting a test pattern through it. The output sequence observed at SDO should be the same as that applied at SDI. Test patterns that apply all four transitions (0->0, 0->1, 1->1 and 1->0) to the bistable elements are often used. An example of such a test pattern is 00110.

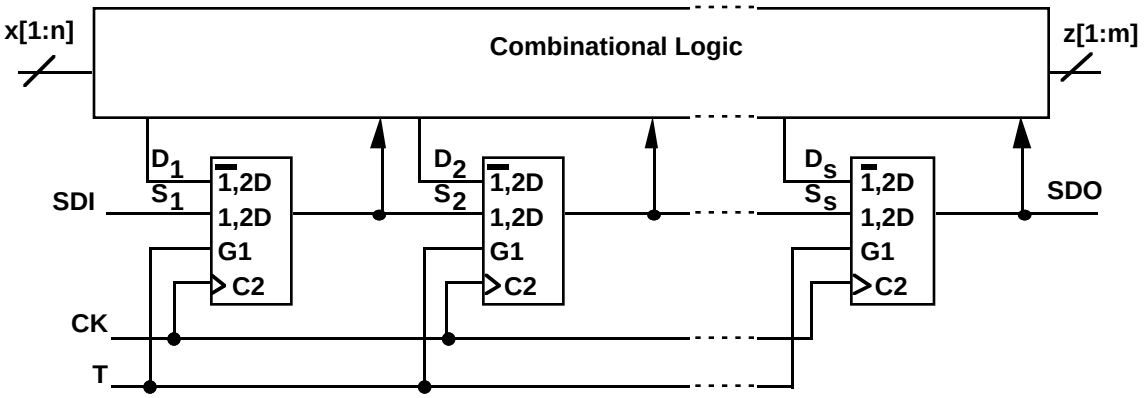


Figure 2-3 MD Flip-Flop Based Scan Architecture.

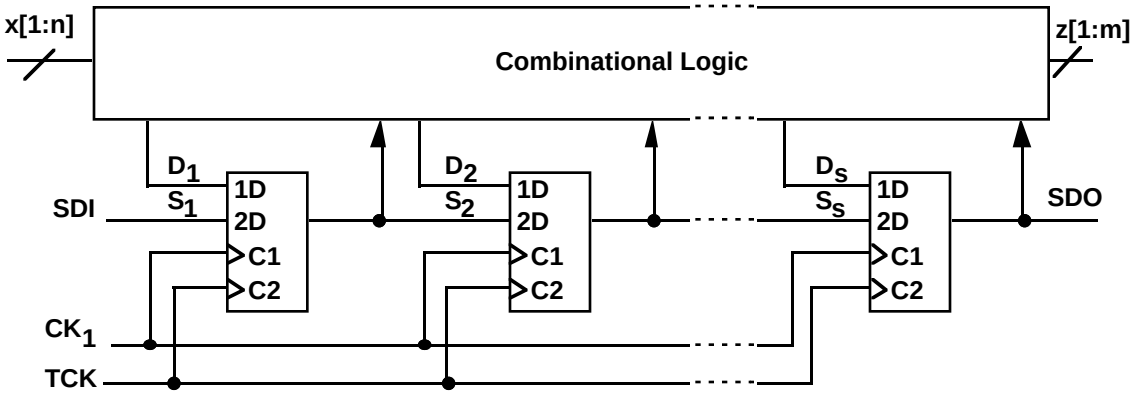


Figure 2-4 TP Flip-Flop Based Scan Architecture.

3. Checking Experiments for Bistable Elements

Flip-flops can have many flow tables, but there is only one primitive flow table (barring isomorphism) for each flip-flop type [McCluskey 86]. 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 a 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, as the shaded MD flip-flop (F_3)in Fig. 3-1, 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, making the task even more difficult.

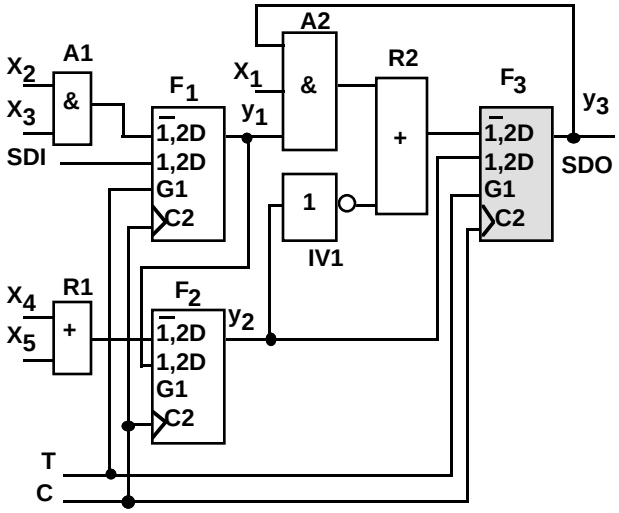


Figure 3-1 Circuit Under Test.

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 needs to be applied to the bistable element under test, and the output of the bistable needs to be observed at a primary output. An example of such an input sequence for MD flip-flop is shown in Fig. 3-2. 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=0, and q=1).

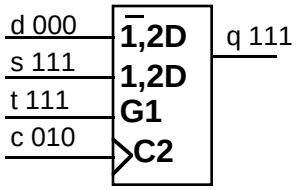


Figure 3-2 A Required Input Sequence for an MD Flip-Flop.

The method for finding a test pattern that would set d=0, s=0, and q=1 is similar to combinational ATPG. In our example, the test pattern would be $X_1X_2X_3X_4X_5y_1y_2y_3$ = 0-----11 would set d, s and q to the desired values (see Fig. 3-3). 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 = 0 and d = 0 after the clock pulse. In our example, this can be done with the test pattern $X_1X_2X_3X_4X_5y_1y_2y_3$ = 0----111 (see Fig. 3-4). 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.

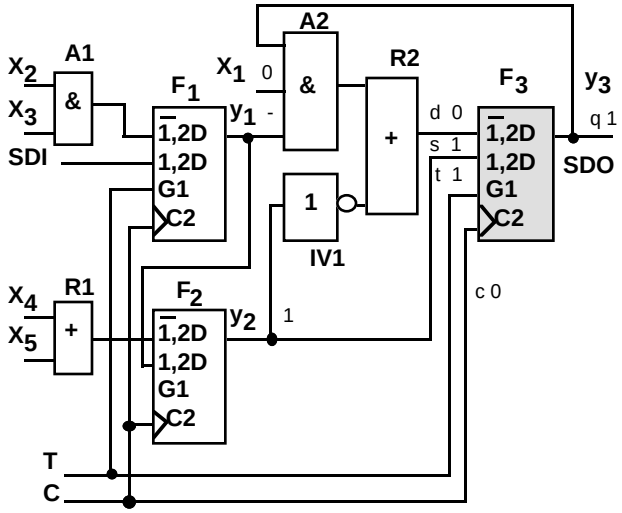


Figure 3-3 Circuit Under Test With Test Pattern.

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. However, since there are only two time frames, the problem is much easier than 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 shift operation. Other required sequences need other types of operations. These operations, called elementary operations, are summarized in Table 3-1. The first two elementary operations are used with sequences for which there is no change in the clock. The third (the one we showed in the example) and fourth are used with sequences in which there is a change in the clock. The last elementary operation is used in conjunction with the other elementary operations when the next flip-flop in the scan chain cannot "capture" the output of the flip-flop under test.

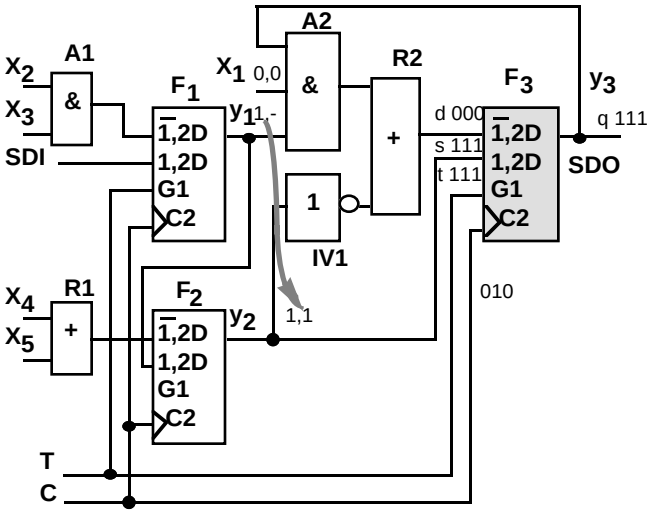


Figure 3-4 Circuit Under Test With Test Pattern.

Table 3-1 Elementary Operations

Operation	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.
Combinational Logic Sensitization	Determine bit values of a test pattern that would sensitize a line in the circuit to a primary output or an input of a bistable element.

Elementary operations can be used to generate test patterns for all the required input sequences [Makar 96]. Therefore, we can define an algorithm for generating test patterns for the bistable elements using the algorithm in Fig. 3-5. 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. In Sec. 2 we introduced four different architectures. The architectures used different bistable elements for scan cells. The different bistable element types have different required sequences, and thus even though their algorithms have the structure in Fig 3-5, each will have a different implementation.

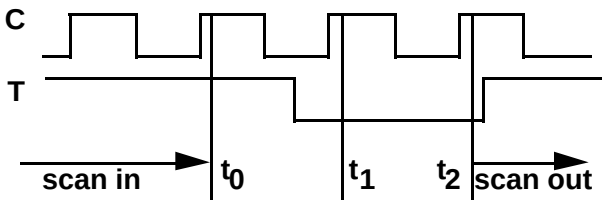
```
for each bistable element
    for each required sequence
        Apply Appropriate Elementary Operation
        Add Test Pattern To Appropriate Table
    }
}
Compact Tables
Print Tables
```

Figure 3-5 Algorithm for ATPG for Bistable Elements.

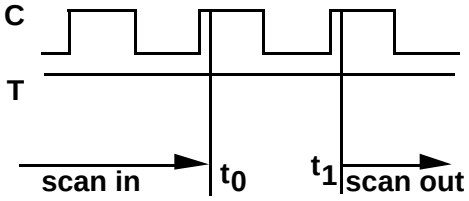
4. Implementation

We implemented our algorithm by modifying an existing stuck-at ATPG program in SIS [Sentovich 92]. This was done by creating elementary functions and using the elementary functions to write procedures for the four different bistable element types used in the scan chain architectures described in Sec. 2.

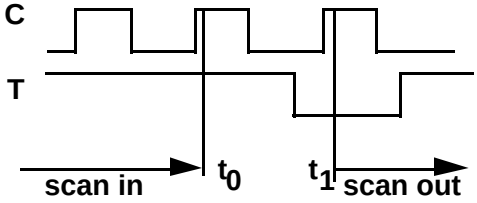
As with most ATPG programs, this program reads a gate level description of the circuit. However, unlike most ATPG programs, the output is not simply a file with test patterns, but rather a set of files with test patterns. The number of test pattern files depends on the scan architecture used. For example, the MD flip-flop architecture creates 6 files. Fig. 4-1 graphically describes how these patterns are applied to the tester, or used in a simulator for verification. For example, for Waveform A0, after a test pattern is scanned in, T is set to 0 for two cycles, set back to 1, and then the flip-flop contents are scanned out. The procedures are described as waveforms rather than text as this is closer to how tester formats are often defined.



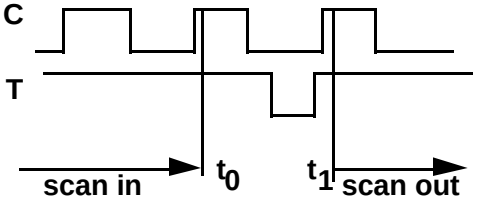
(a) Waveform A0



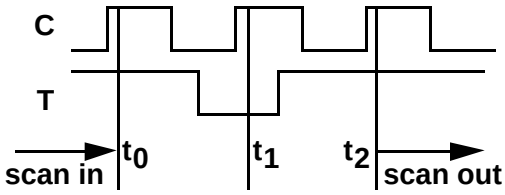
(b) Waveform A1



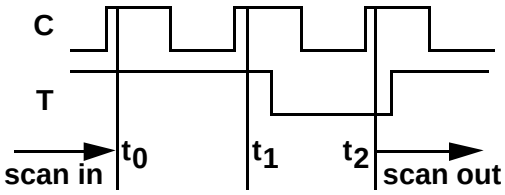
(c) Waveform B0



(d) Waveform B1



(e) Waveform C0



(f) Waveform C1

Figure 4.2-1 Waveforms for Applying Test Patterns for MD Flip-Flop Scan Chain.

5. 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, [Sucar 89] and [Chandra 93], 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 [Kielkowski 94].

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 [Ma 95]. 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 die, and selecting an appropriate threshold that would detect as many faulty circuits as possible without discarding many good ones [Hawkins 89] and [Perry 92]. For our simulations, the current limit is determined by plotting the maximum observed current for each fault, and selecting an appropriate threshold from the graph.

In Section 5.1, we present simulation results of an MD flip-flop comparing traditional tests with checking experiment based tests. In Section 5.2, we present fault simulation results for a three bit binary counter, again comparing traditional test techniques with our new approach. Results in both cases show that we detect many faults missed by traditional approaches. In Section 5.2, we present ATPG results for all the ISCAS 89 circuits. We compare the length of our tests with the length of traditional stuck-at tests. The test lengths increase at the same rate, indicating that not many more test patterns are needed for large circuits.

5.1 MD Flip-Flop Fault Simulation.

Four 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 other two tests are the MD flip-flop checking experiment (Table 3-2) and the checking experiment for the MD-latch in a scan chain. The flip-flop implementation used for the simulation is shown in Fig. 5.1-1. This implementation is selected because it is a commonly used structure.

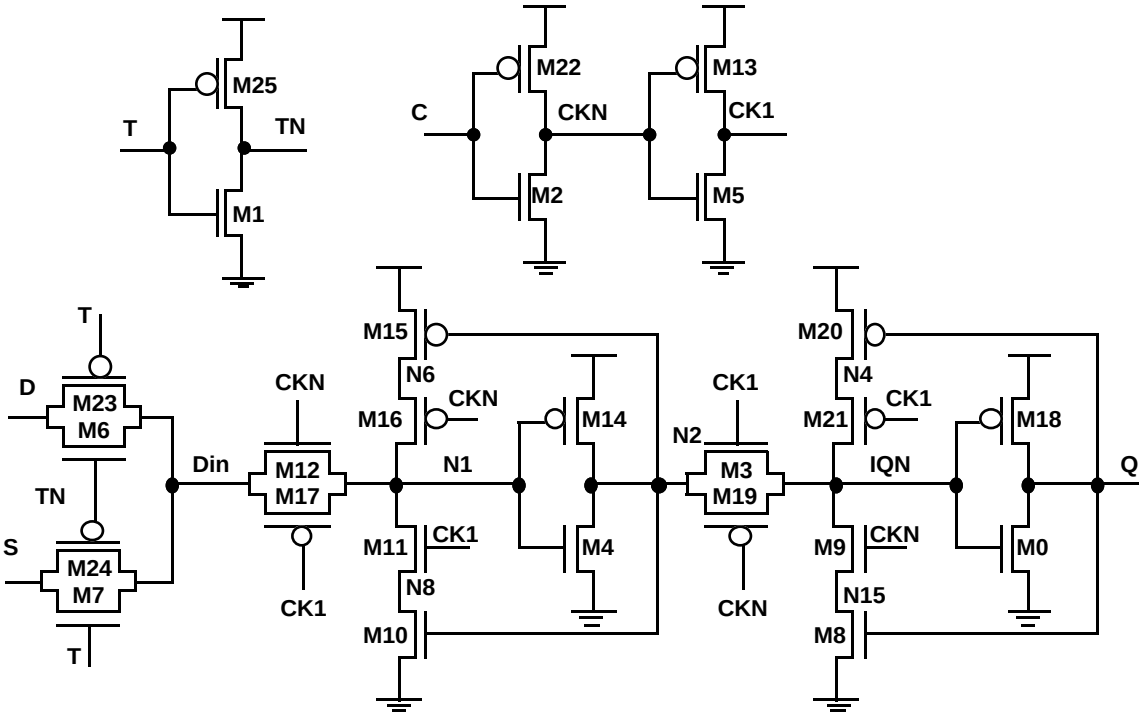


Figure 5.1-1 MD Flip-Flop Implementation Used in Simulation.

Fig. 5.1-2 shows the distribution of maximum IDDQ values for circuits with faults injected. IDDQf refers to the IDDQ value of the circuit with a fault injected, and IDDQg the value of IDDQ of the fault free circuit. The graph shows a sharp rise in IDDQf / IDDQg from 1 to 100,000. Such a sharp break in the graph makes for a good IDDQ threshold. Therefore, the IDDQ threshold selected is 100,000 times the IDDQ current of the fault-free circuit. The maximum IDDQ value for the fault-free circuit was 1 nA, thus the threshold selected was 100 uA.

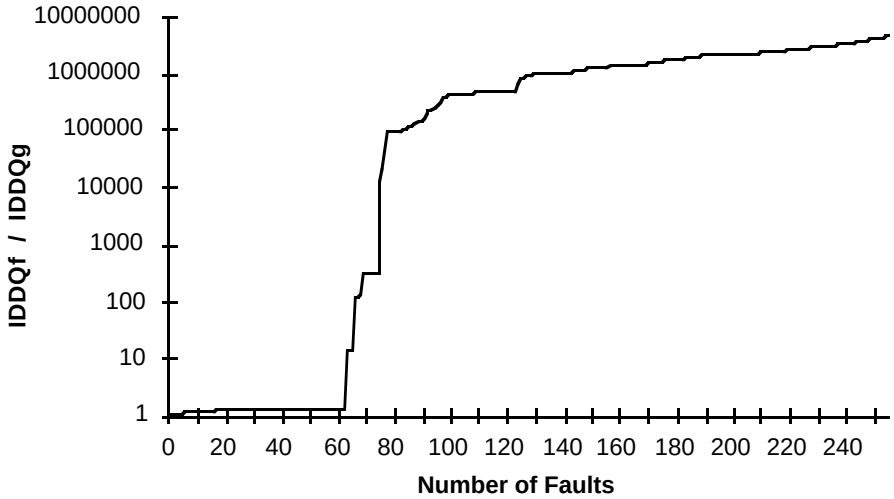


Figure 5.1-2 Current Distribution Graph for MD Flip-Flop (IDDQg = 1 nA).

The results of the simulations are shown in Table 5.1-1. From the table, there are 19 faults that were not detected by the checking experiment. These faults are shown graphically in Fig. 5.1-3. The table also shows that the pin fault test misses ten faults that are detected by the checking experiment. These faults are shown in Fig. 5.1-4. In these figures white ovals indicate stuck-open or open-interconnect faults, black ovals indicate SON faults, and thick black lines indicate shorted-interconnect faults. All short-to-power and short-to-ground faults are detected by all tests.

The faults missed by the checking experiment fall into two groups . The first group of faults missed by the checking experiment is the stuck-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 to the input of the flip-flop may be able to detect these faults. The other group of faults missed by the checking experiment, the stuck-ons and

Table 5.1-1 Number of Faults Detected in MD Flip-Flop (Total Faults = 256).

	Boolean and IDDQ	Boolean Alone (100 ns and 10 ms)	Boolean Alone (100 ns, 10 ms)	IDDQ Alone
Traditional Test	212	167	(145,166)	155
Pin Fault Test	227	184	(162,183)	161
MD Flip-Flop Checking Exp.	237	207	(186,204)	182
Scan MD Flip-Flop Checking Exp.	237	206	(184,204)	181

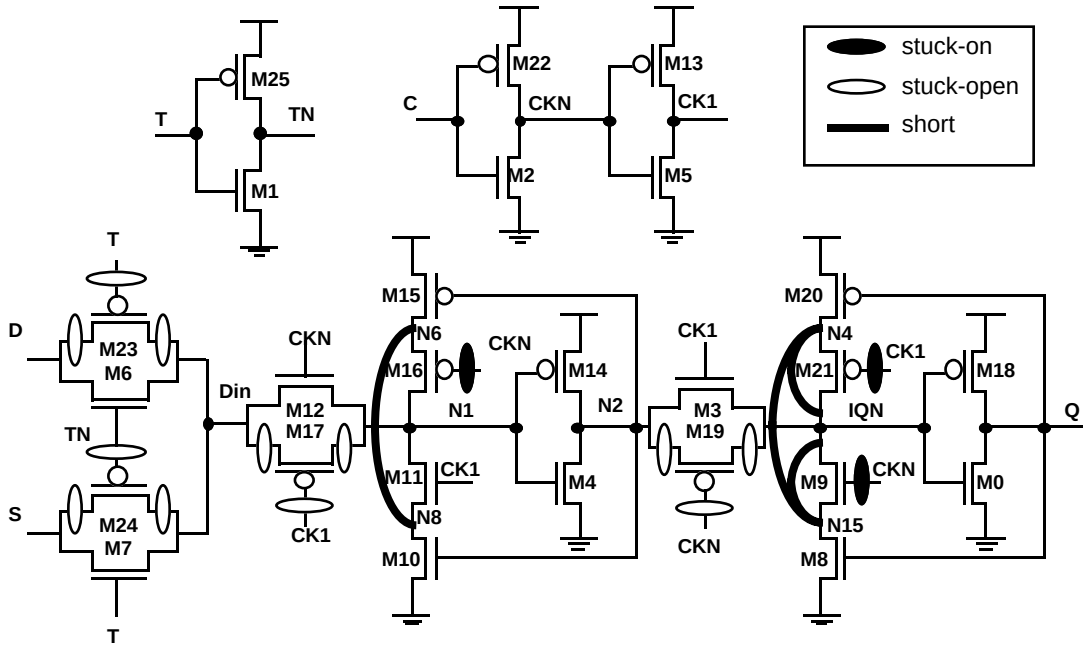


Figure 5.1-3 Faults Missed by Checking Experiment of MD Flip-Flop (19 of them).

shorted-interconnects, 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.

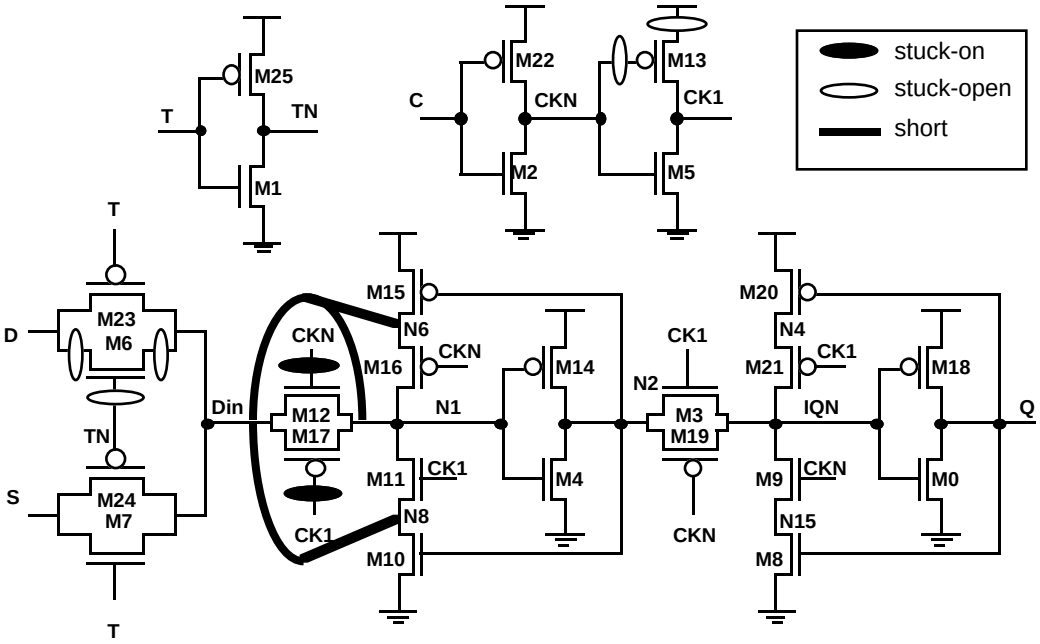


Figure 5.1-4 Faults Missed by Pin Fault Test Detected by Checking Experiment of MD Flip-Flop (10 of them).

5.2 Circuits Using MD Flip-Flop

In this section we present some results from running our new ATPG program described in Sec. 4. First, we present results for a small circuit, a three-bit binary counter. A binary counter is chosen because it is a commonly used circuit. A three-bit version is picked so that the circuit is general enough, while at the same time small enough to allow for HSpice fault simulation. As in section 5.1, HSpice is used to perform fault simulation using CrossCheck models, to compare the effectiveness of different tests. Even though it would be difficult to fault simulate the ISCAS-89 benchmark circuits [Brglez 89] using HSpice, we generate test patterns for them, and compare their size with the size of stuck-at test patterns. Results indicate that the size of our test increases with circuit size at a similar rate as the stuck-at test.

Two tests were generated for the three-bit counter shown in Fig. 5.2-1. The first test, a traditional test, consists of five test patterns: four to give 100 percent stuck-at coverage for the combinational logic, and the 01100 test pattern for testing the flip-flops. The stuck-at test patterns

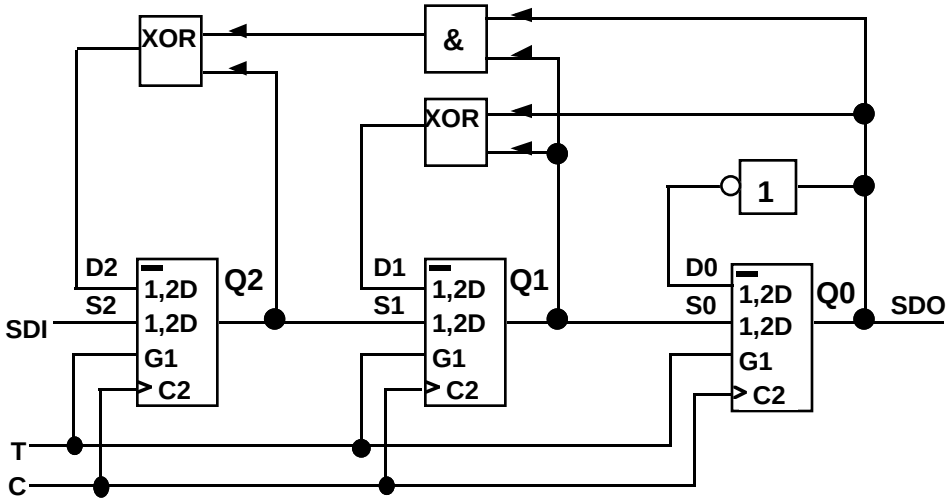


Figure 5.2-1 Circuit for a Three-Bit Binary Counter.

are shown in Table 5.2-1. The second test was generated by our ATPG program for MD flip-flops described in Section 4. The test consists of 82 test patterns. Table 5.2-2 shows a summary of this test. Each MD flip-flop requires 80 sub-sequences (see Table 3-5) to form a checking experiment. Since there are 3 flip-flops, we have 240 sub-sequences. The program generated test patterns for 133 of them. For the other 107, the program proved that no test pattern could be generated, due to the circuit structure. For example, looking at the Q0 flip-flop, D0 will always be the opposite value of Q0, which implies that D0 cannot hold the same value for two cycles, a requirement for many of the sub-sequences needed to form a checking experiment for flip-flop Q0. The 133 test patterns were compacted to the 82 test patterns in Table 5.2-3. The size of our test is a lot larger than the stuck-at test. This is mainly due to the small size of combinational logic in the circuit. Later in this section, we show that the test size difference decreases with larger circuits.

The test patterns for HSpice must be scanned in before applying them to the internal logic. Each test pattern for the combinational logic requires three cycles to be scanned in, one cycle to capture the output of the combinational logic, and three cycles to scan out the flip-flop values. Since a new test pattern can be scanned in while the flip-flop values are scanned out, each test pattern requires 4 cycles (three for scanning the test pattern and one for capturing the output of the

Table 5.2-1 Stuck-At Test Patterns for Three-Bit Binary Counter.

Pattern No.	$Q_2Q_1Q_0$
1	1 1 1
2	0 1 1
3	1 0 1
4	1 1 0

combinational logic). On the other hand, test patterns from our ATPG tool require four or five cycles.

Table 5.2-2 Results of ATPG for Three-Bit Binary Counter.

Number of Sub-sequences	240
Number of Sub-sequences With No Test Pattern	107
Number of Aborted Sub-sequences	0
Number of Sub-sequences With Test Pattern	133
Number of Compacted Test Patterns	83

Both test sets were simulated in HSpice using the fault models described in Section 5.1. The results are summarized in Table 5.2-3. The checking experiment detected 45 faults that were not detected by the stuck-at test set. This small example shows that a 100 percent stuck-at test (i.e., a test generated to detect all stuck-at faults in the circuit) can leave many real flip-flop faults undetected.

Table 5.2-3 shows that the checking experiment missed a large number of faults. The large number of undetected faults is a direct result of the large number (107) of sub-sequences for which no test pattern can be generated. Another reason for the large number of undetected faults is that we only simulated the 100 ns case because of the size of the circuit. Therefore, we could not make IDDQ measurements. As shown in Section 5.1, IDDQ measurement detects many faults not detected by boolean testing.

Table 5.2-3 Summary of Simulation Results for Three-Bit Binary Counter.

Faults Injected	651
Faults Not Detected by Traditional Method	393
Faults Not Detected by Checking Experiment	348
Faults Detected by Checking Exp. Missed by Traditional	45

One practical concern with testing chips is the size of the test being applied. To address this issue, we generated test patterns for the ISCAS 89 benchmark circuits for all four architectures, and compared them to the stuck-at test lengths. Table 5.2-4 shows the number of vectors for all the ISCAS 89 circuits for each architecture, and for the stuck-at tests. The name of the ISCAS 89 circuits indicates the number of lines in the circuit. This is directly related to the size of the circuit. The number of test patterns for the LSSD architecture is always the smallest of our tests, and the number of test patterns for the MD flip-flop architecture is always the largest.

To compare our test size with the test size of the stuck-at test, we calculate the ratio of the size of our tests to the size of the stuck-at tests. These ratios are shown in Table 5.2-5. The numbers in this table were calculated by dividing the number of test patterns for the bistable

elements by the number of stuck-at test patterns. Since the ratios do not show an increase with circuit size, we conclude that the size of our test will not be a problem with large circuits.

Table 5.2-4 Number of Test Patterns for Different Tests.

Circuit	MD-Latch	LSSD	MD Flip-Flop	TP Flip-Flop	Stuck-At
S27	29	19	118	52	14
S298	71	47	356	162	66
S344	97	62	395	165	65
S349	91	61	375	158	66
S382	135	85	578	244	74
S386	49	31	282	114	88
S400	133	85	576	242	71
S444	97	65	487	212	80
S510	65	43	306	149	78
S526	106	68	572	257	139
S641	187	128	578	257	124
S713	123	84	572	255	133
S820	54	37	346	144	161
S832	54	37	349	148	171
S1196	126	80	440	233	201
S1423	341	216	1640	756	218
S1488	68	45	421	179	247
S1494	68	45	422	180	243
S5378	571	336	2139	1023	700

Table 5.2-5 Number of Test Patterns Divided by Stuck-At Test Length.

Circuit	MD-Latch	LSSD	MD Flip-Flop	TP Flip-Flop	Stuck-At
S27	2.07	1.36	8.43	3.71	1.00
S298	1.08	0.71	5.39	2.45	1.00
S344	1.49	0.95	6.08	2.54	1.00
S349	1.38	0.92	5.68	2.39	1.00
S382	1.82	1.15	7.81	3.30	1.00
S386	0.56	0.35	3.20	1.30	1.00
S400	1.87	1.20	8.11	3.41	1.00
S444	1.21	0.81	6.09	2.65	1.00
S510	0.83	0.55	3.92	1.91	1.00
S526	0.76	0.49	4.12	1.85	1.00
S641	1.51	1.03	4.66	2.07	1.00
S713	0.92	0.63	4.30	1.92	1.00
S820	0.34	0.23	2.15	0.89	1.00
S832	0.32	0.22	2.04	0.87	1.00
S1196	0.63	0.40	2.19	1.16	1.00
S1423	1.56	0.99	7.52	3.47	1.00
S1488	0.28	0.18	1.70	0.72	1.00
S1494	0.28	0.19	1.74	0.74	1.00
S5378	0.82	0.48	3.06	1.46	1.00

Conclusions

We presented a new approach for testing bistable elements in digital circuits. Traditional approaches for testing bistable elements in a scan chain involve shifting in a sequence of zeroes and ones. We showed that this approach misses many faults in the circuit. These faults may affect normal circuit operation. 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. Analysis of faults inside bistable elements [Al-Assadi 93] has shown that some faults inside the bistable elements cannot be mapped to functional fault models. This implies that transistor level test generation is needed to target the specific faults. However, the checking experiment will detect these faults without special analysis since it detects all the faults inside the bistable element.

Our implementation of the test generator was based on modifying an existing stuck-at test tool. This was done to illustrate that current tools can be easily adapted to include tests for bistable elements. The implementation was done in a hierarchical fashion, where elementary functions were used to implement common features needed for the different architectures. This not only made it easier to debug problems, but also makes it easier to add tests for new bistable element types in the future. The implementation currently includes test generators for the four different implementations discussed in Section 2.

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. Though the size of our test was considerably larger than that of the stuck-at test for the small binary counter, we showed that the test size increases with circuit size by about the same rate as the test for stuck-at faults. 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.

Even though we have shown that the test length is not significantly larger than stuck-at tests for combinational logic, we will be considering some methods to reduce the number of test patterns. One approach we are considering is to remove parts of the checking experiments that do not detect any defects for a given bistable element implementation. This would require accurate fault simulation of the bistable elements in the library. Even though our test patterns target the bistable elements, they are likely to detect some of the defects in the combinational logic as well. Another approach we are considering is to use our test patterns as functional vectors for stuck-at ATPG tool. This will reduce the overall number of test patterns. We also plan to try our test

patterns on a test chip to see how many defective dice our test can detect. This experiment should confirm our assertion that our tests detect real defects that are normally missed by traditional tests.

Acknowledgment

The authors would like to thank Jonathan Chang for his 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

[Al-Assadi 93] 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.

[Brglez et. al. 89] Brglez, F., D. Bryan and K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuit," *IEEE International Symposium on Circuits and Systems*, pp. 1929-1934, 1989.

[Chandra 93] Chandra, S., K. Pierce, G. Srinath, H. Sucar and V. Kulkarni, "CrossCheck: An Innovative Testability Solution," *IEEE Design and Test of Computers*, Vol. 10, No. 2, pp. 56-67, June, 1993.

[Eichelberger 77] Eichelberger, E.B., and T.W. Williams, "A Logic Design Structure for LSI Testability," *Proc. 14th Des. Autom. Conf.*, New Orleans, LA. pp. 462-468, June 20-22, 1977.

[Hawkins 89] Hawkins, CF., "Quiescent Power Supply Current Measurement for CMOS IC Defect Detection," *IEEE Trans. on Industrial Electronics*, pp. 211-218, May, 1989.

[Hennie 64] 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.

[Kielkowski 94] Kielkowski, R., *Inside SPICE Overcoming the Obstacles of Circuit Simulation*, McGraw Hill, USA, 1994.

[Kubuo 68] Kubo, H., "A Procedure for Generating Test Sequences to Detect Sequential Circuit Failures," NEC Res. and Dev., No. 12, October, 1968.

[Larrabee 89] Larrabee, T., "Test Pattern Generation Using Boolean Satisfiability," *IEEE Trans. on CAD*, pp. 4-15, January, 1989.

[Lee 90] Lee, K.J. and M.A. Breuer, "A Universal Test Sequence for CMOS Scan Registers," *IEEE Custom Integrated Circuits Conference*, pp. 28.5.1-28.5.4, 1990.

[Ma 95] Ma, S., P. Franco and E.J. McCluskey, "An Experimental Chip to Evaluated Test Techniques, Experimental Results," *Proceedings ITC*, pp. 663-672, 1995.

[Makar 95] Makar, S.R. and E.J. McCluskey, "Functional Tests for Scan Chain Latches," *Proceedings ITC*, pp. 606-615, 1995.

[Makar 96] Makar, S.R. and E.J. McCluskey, "Checking Experiments for Scan Chain Latches and Flip-Flops," CRC Technical Report, in progress.

[McCluskey 86] McCluskey, E.J., *Logic Design Principles*, Prentice-Hall, New Jersey, 1986.

[Mourad 90] Mourad, S., "Sequential Circuit Testing," *COMPCON*, p 449-454, 1990.

[Muth 76] Muth, P., "A Nine-Valued Circuit Model for Test Generation," *IEEE Trans. on Computers*, Vol. C-25, No. 6, pp. 630-636, 1976.

[Perry 92] Perry, R., "IDDQ Testing in CMOS Digital ASICs - Putting It All Together," *Proceedings ITC*, pp. 151-157, 1992.

[Putzolu 71] Putzolu, G.R. and J.P. Roth, “A Heuristic Algorithm for the Testing of Asynchronous Circuits,” *IEEE Trans. on Computers*, Vol. C-20, No. 6, pp. 639-647, June, 1971.

[[Reddy 86] 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.

[Sentovich et. al. 92] Sentovich, E.M., K.J. Singh, L. Lavagno, C. Moon, R. Mrgai, A. Saldanha, H. Savoj, P. R. Stephan, R.K. Brayton, A.S. Vincentelli, “SIS A system for Sequential Circuit Synthesis,” *Electronics Research Lab* Memorandum, No. UCB/ERL M92/41, 1992.

[Sucar 89] Sucar, H., “High Performance Test Generation for Accurate Defect Models in CMOS Gate Array Technology,” *ICCAD*, pp. 166-169, 1989.

[Williams 73] Williams, M.J. and J.B. Angel, “Enhancing Testability of Large Scale Integrated Circuits via Test Points and Additional Logic,” *IEEE Trans. on Computers*, C-22, No. 1, pp. 46-60, January, 1973.