Finite State Machine Synthesis for At-Speed Oscillation Testability

Katherine Shu-Min Li[1], Chung Len Lee[1], Tagin Jiang[2], Chauchin Su[3], Jwu E. Chen[4]
[1]Department of Electronics Engineering, National Chiao Tung University, Hsichu, Taiwan
[2]Via Technologies, Inc.
[3]Department of Electronic Control, National Chiao Tung University, Hsichu, Taiwan
[4]Department of Electrical Engineering, National Central University, Chungli, Taiwan

ABSTRACT

In this paper, we propose an *oscillation-based* test methodology for sequential testing. This approach provides many advantages over traditional methods. (1) It is *at-speed* testing, which makes *delay-inducing defects* detectable. (2) The ATPG is much easier, and the *test set* is usually smaller. (3) There is no need to store output responses, which greatly reduces the communication bandwidth between the Automatic Test Equipment (ATE) and Circuit under Test (CUT). We provide a register design that supports the oscillation test, and give an effective algorithm for oscillation test generation. Experimental results on MCNC benchmarks show that the proposed test method achieves high fault coverage with smaller number of test vectors.

1. Introduction

Decreasing feature sizes and increasing clock speeds have combined to alter the defect effects dramatically. Recent evidence indicates that delay-inducing defects can no longer be ignored nor go untested [1-2]. For circuits designed with 130nm or more advanced technologies, the transition fault is considered essential to achieve the acceptable defect level. The detection of delay fault requires at-speed test techniques, which create signal transitions to be captured at normal speed. In the past, it was typically accomplished with functional patterns, but it was undesirable mainly due to the cost consideration. Scan-based test techniques [3-4] offer a viable alternative for at-speed testing. However, there are many complicating factors when moving from relatively slow scan-based tests for stuck-at faults to testing for delay faults. As to design methodologies such as multiple clock domains, mixed negative and positive edge clocking, and so on, all pose challenges to the implementation of successful and high coverage delay tests. The cost associated with such design methodologies is also an ever increasingly important issue.

We propose an *oscillation-based* test methodology for sequential testing in this paper. This approach provides three major advantages over traditional scan-based approaches. (1) In this architecture, testing is conducted at-speed, which makes delay-inducing defects detectable. This is due to that the oscillation test is triggered by system clock and thus operates at normal speed. (2) Test vectors can be derived directly from the finite-state machine (FSM) model in our Oscillation Test Pattern Generation (OTPG) algorithm, and it greatly simplifies the ATPG process accordingly. (3) Our method does not need complex test clocks, which is required for two-pattern tests used in transitional delay tests. (4) The correctness of CUT is determined by simply observing whether there are oscillation signals in the outputs, and there is no need to store and analyze output responses. Besides, the number of vectors is roughly the same as scan tests. Thus, the communication bandwidth between the ATE and CUT is greatly reduced, which partly solves the problem of test data compression in SOC testing.

Oscillation based test is an efficient and effective method to detect faults in a circuit or a device [5-6]. An oscillation ring is a closed loop with an odd number of signal inversions. If the CUT is fault-free, an oscillation signal will appear on the ring. Otherwise, the CUT is deemed faulty. Recently oscillation ring test is applied for system-level interconnects [7].

In order to conduct the oscillation test, the state-holding elements must be modified to generate oscillation signals in test mode. In this paper, we develop a *Modified State Register* (MSR) cell for this purpose, and give an algorithm to generate tests with the help of MSR cells. The proposed MSR design requires extra silicon area. However, in deep submicron designs, silicon area is no longer the major issue. Other issues, including *delay fault* and *soft fault testability*, *low-power testing*, etc., become the more important concerns. For example, Intel proposes a scan-cell design targeted for soft faults [8]. This cell-level design uses 1.08X-1.24X area with power overhead of 2.02X-2.26X, while chip-level design suffers from power overhead by 4.0X-5.0X [8]. The proposed MSR cell can be combined with other register designs to achieve highly testable and reliable systems.

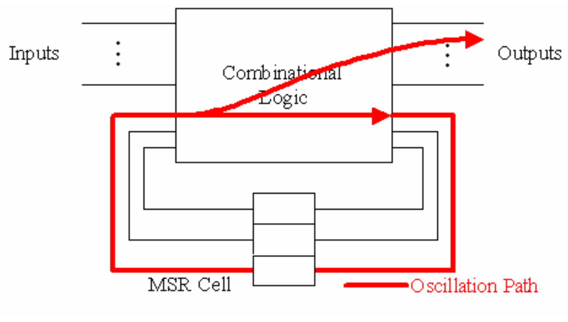
Experimental results on MCNC benchmark circuits show that the proposed oscillation test method achieves high fault coverage with smaller number of test vectors.

The remaining sections are organized as follows. In Section 2, we introduce the proposed Oscillation Test architecture and MSR cell designs for both asynchronous and synchronous circuits. Section 3 gives Oscillation Test Pattern Generation (OTPG) algorithm. Experimental results are shown in Section 4, and some brief conclusions are in Section 5.

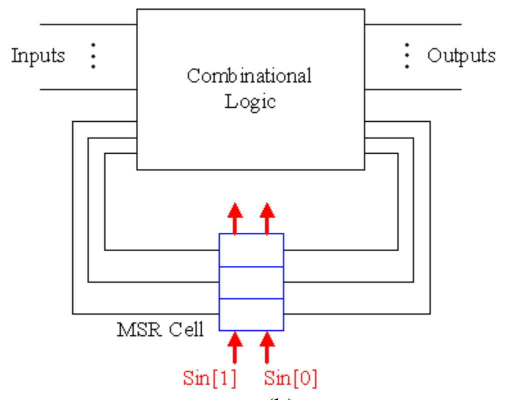
2. Oscillation Test for Sequential Circuits

2.1 Oscillation Ring Test Architecture

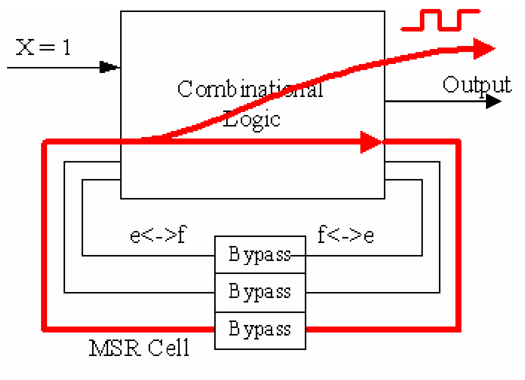
The oscillation ring test architecture for sequential circuits is shown in Figure 1. In this architecture, we replace the flip-flops by MSR cells. In the normal mode operation, the MSR cells work as state-holding elements. In the oscillation test mode, MSR cells transform the target sequential circuits into asynchronous circuits with odd-inversion feedback paths, and oscillation signals show on these loops (rings) accordingly.



(a)



(b)



(c)

Figure 1. Oscillation test architecture for sequential circuits: (a) Oscillation Rings, (b) MSR states are controlled through scans, and (c) Oscillation Test is controlled by the system clock.

With appropriate inputs, oscillation signals can be propagated to at least one primary output through some sensitized paths in Figure 1(a). Stuck-at faults on wires passed by oscillation signals will stop these oscillation signals; while delay (transition) faults will change the oscillation frequency. The faults are detected by observing the oscillation signals in the primary outputs.

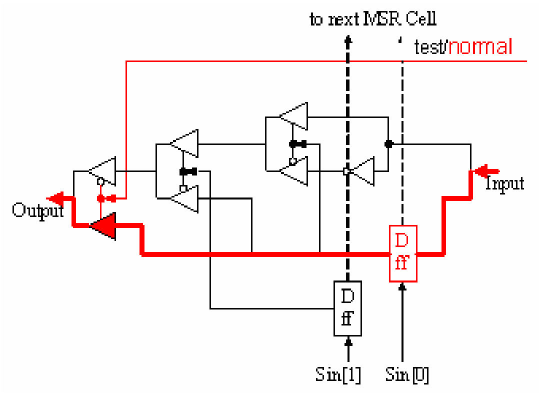
In order to construct oscillation rings in sequential circuits, we need to set up appropriate connections in MSR cells. Figure 1(b) shows how to set the states in MSR cells. The control signals for each MSR cell are fed to the cell through the scan paths.

It is usually difficult to implement the asynchronous test architecture. Whenever there are multiple oscillation signals, there is always a race problem. To solve this problem, we may use the system clock to control the feedback paths, which makes the design synchronous, as shown in Figure 1(c). The circuit is forced to move between states *e* and *f*, whose outputs are 0 and 1, respectively, when the input *X* is held at *X*=1. As a result, we can see that the output changes every cycle.

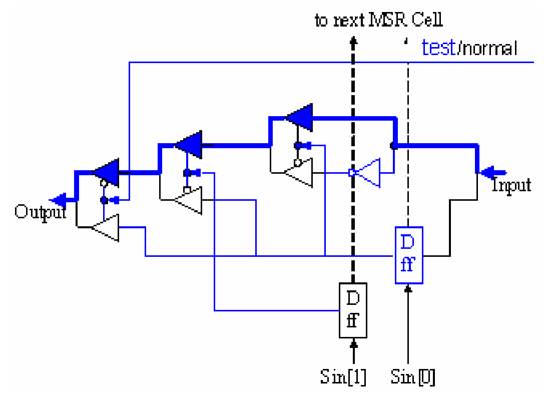
2.2 Modified State Register (MSR) Design

2.2.1 MSR Design for Asynchronous Test

The MSR cell design for asynchronous oscillation test is shown in Figure 2, and the control states for the MSR cells are shown in Figure 3.



(a)



(b)

Figure 2. MSR cells for asynchronous oscillation test: (a) normal mode, and (b) oscillation test mode.

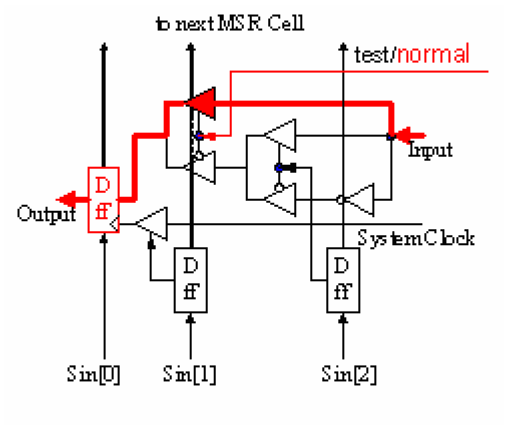
Under normal operation in Figure 2(a), the D-Type Flip Flops (DFFs) connected to Sin[0] are used as state-holding elements. Under the oscillation test mode in Figure 2(b), an MSR cells operate in four states: Hold 0, Hold 1, INV and Bypass. Hold 0 and Hold 1 provide steady output values of 0 and 1, respectively. INV and Bypass are used to set up odd-inversion loops to generate oscillation signals. A loop (ring) consists of two paths: one forward path in the combinational circuit, and a feedback path passing an MSR cell. If the number of signal inversion in the forward path is odd, the MSR cell is set to the "Bypass" state; otherwise, it is set to the "INV" state.

$S_{in}[1]$	$S_{in}[0]$	Operation
0	0	Hold 0
0	1	Hold 1
1	0	INV
1	1	Bypass

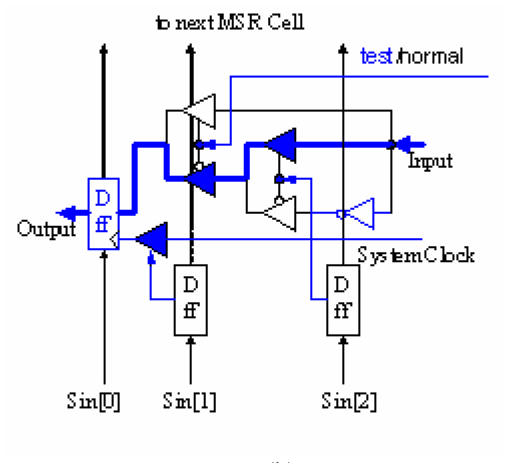
Figure 3. Control state table of an MSR cell for asynchronous sequential circuit test.

2.2.2 MSR Design for Synchronous Sequential Circuits

In order to avoid the race conditions caused by the asynchronous test, we use system clock to sample the oscillation signals, as shown in Figure 1(c).



(a)



(b)

Figure 4. MSR cells for synchronous oscillation test: (a) normal mode, (b) oscillation test mode.

An MSR cell's design and its operations for the synchronous oscillation test are illustrated in Figure 4, and the control states for MSR cells are given in Figure 5.

$S_{in}[2]$	$S_{in}[1]$	$S_{in}[0]$	Operation
-	0	0	Hold 0
-	0	1	Hold 1
0	1	-	INV
1	1	-	Bypass

Figure 5. Control state table of MSR cells for synchronous sequential circuits test.

3. Synchronous Oscillation Ring Test

The synchronous oscillation ring test is preferred for several reasons. The most important advantage is that it avoids race problems, which is very difficult to handle in the asynchronous approach. Secondly, it also simplifies the ATPG process. The test patterns can be obtained directly from the FSM model. The drawback of this approach is that an MSR cell is significant larger. This large hardware overhead can be partly offset if we can restrict the number of operations required in an MSR cell, and this can be achieved through state assignment for the given FSM. In the remaining part of the paper, we shall concentrate on the synchronous oscillation test.

3.1 Constructing Oscillation Signals from FSM

An example on how to find test patterns from an FSM model is given in Figure 6, which shows the state transition and output table of an FSM. The output table gives the candidates for oscillating outputs. For example, when the primary input *X* is held at 1 (*X*=1) and the FSM is in either state *e* or *f*, the FSM moves back and forth between these two states. Since the outputs corresponding to states *e* and *f* with *X*=1 are 0 and 1, respectively, we shall see oscillating signals at the output. This oscillation condition is shown in Figure 1(c), in which the least significant bit (LSB) of the state vector is an oscillating signal.

Present State	Next State		Output	
	X=0	X=1	X=0	X=1
a 000	a 000	c 010	1	0
b 001	d 011	b 001	1	0
c 010	f 101	d 011	1	1
d 011	c 010	a 000	0	1
e 100	e 100	f 101	0	0
f 101	b 001	e 100	1	1

Figure 6. State transition and output table of an FSM.

In the above example, an oscillation signal is generated without using MSR cells. This is achievable when both the next states and outputs of a state pair are alternating. Unfortunately, no other state pairs satisfy the oscillation condition. We shall use MSR cells to force state pairs to alternating if only their corresponding outputs are different. All the candidate state pairs are listed below. With *X* = 0, the following state pairs generate the opposite output values: (*a*, *d*), (*a*, *e*), (*b*, *d*), (*b*, *e*), (*c*, *d*), (*c*, *e*), (*f*, *d*), (*f*, *e*). With *X*=1, the possible choices include: (*a*, c), (*a*, *d*), (*a*, *f*), (*b*, *c*), (*b*, *d*), (*b*, *f*), (*e*, c), (*e*, *d*), (*e*, *f*).

3.2 MSR State Transition Algorithm (MSR STA Algorithm)

In order to generate oscillation signals in the test mode, we need to change the next state functions for the selected state pair with the help of MSR cells. An example is shown in Figure 7, which modifies the state transition table of Figure 6 in the test mode to produce oscillation signals.

For example, since state pair (*b*, *e*) in Figure 6 produces different outputs when *X*=0, it is a candidate for generating oscillation signals. To do this, in the test mode we need to change the next states of (*b*, *e*) from (*d*, *e*) to (*e*, *b*) when *X*=0, as indicated in Figure 7.

Present State	Next State		Output	
	X=0	X=1	X=0	X=1
a 000	a 001 e	c 010	1	0
b 001	d 011	b 001	1	0
c 010	f 101	d 011	1	1
d 011	c 010	a 000	0	1
e 100	e 100 b	f 101	0	0
f 101	b 001	e 100	1	1

Figure 7. Modified state transition table.

Bit Transition	OP Value
0 -> 0	Low
0 -> 1	Rising
1 -> 0	Falling
1 -> 1	High

(a)

OP Value		2nd Operand			
		L	H	R	F
1st Operand	L	Bypass	INV	Hold 0	Fail
	H	INV	Bypass	Fail	Hold 1
	R	Hold 0	Fail	INV	Bypass
	F	Fail	Hold 1	Bypass	INV

(b)

Figure 8. (a) Truth table of a state bit transition, (b) Operation table of an MSR cell state.

The modification of next state functions in the test mode can be achieved by setting MSR cells to appropriate states. We present an algorithm to select the MSR states in this section. Two tables are used in this algorithm: (1) Truth table of a state bit transition (Figure 8(a)), and (2) Operation table of an MSR cell state (Figure 8(b)).

In Figure 8(a), the state bit transition shows the bit change between current state and next state. There are four operation definitions: (1) when both current and next states are "0", the operation value is "Low"; (2) when both current and next states are changed from "0" to "1", the operation value is "Rising"; (3) when both current and next states are changed from "1" to "0", the operation value is "Falling"; (4) when both current and next states are "1", the operation value is "High".

In Figure 8(b), the operation table of an MSR cell state defines the state transition relationship between normal next state in normal mode (i.e. operation value 1) and alternate next state in test mode (i.e. operation value 2). "Fail" state is not defined in the MSR cell due to conflicts between two operation values. Please note this table is symmetric due to binary commutative characteristic, and the inverse diagonal is full of "Fail" entries. As to how the operation entries are derived, we show in the following paragraphs.

In Figure 8(b), there are four types of operation definitions in the MSR operation table. The first type in Figure 9 is the "Bypass" state in the MSR cell. It means that the MSR cell's output is the same as its input. As shown in Figure 9(a), two state bits in Present State (PS) are "0", and two state bits in Next State (NS) are also "0" in the alternate state pair. According to the truth table of a state bit, both operation values are "Low", which leads to "Bypass" state. The "Bypass" state satisfies the state transition condition that present and next states are the same. Another example for "Bypass" is in Figure 9(b). The difference between Figure 9(a) and 9(b) is that there are oscillation signals in Figure 9(b) since one PS bit is "0" while the other is "1". In summary, when two operation values are {L, L}, {H, H}, or {R, F}, MSR Cell State is set to "Bypass".

PS	NS	OP		MSR Cell		PS
0	0	L	=>	Bypass	=>	0
0	0	L	=>		=>	0

(a)

PS	NS	OP		MSR Cell		PS
0	1	R	=>	Bypass	=>	1
1	0	F	=>		=>	0

(b)

Figure 9. Operation values of (a) {L, L}, (b) {R, F}.

The second type in Figure 9 is the "INV" state in an MSR cell. It means that the MSR cell's output is the complement of its input. As in Figure 9(c), two state bits in Present State (PS) are "0" and two state bits in Next State (NS) are "1" in the alternate state pair. To achieve this, the MSR cell state must be "INV". According to the truth table of a state bit in Figure 8(a), both operation values are "Rising". Another example for "INV" is in Figure 9(d). The difference between Figure 9(c) and 9(d) is that there is oscillation signals in Figure 9(d) since current bits (PS) are "0" "1" alternate. In summary, when two operation values are three types of {R, R}, {F, F} or {H, L}, MSR Cell State is set to "INV".

PS	NS	OP		MSR Cell		PS
0	1	R	=>	INV	=>	0
0	1	R	=>		=>	0

(c)

PS	NS	OP		MSR Cell		PS
1	1	H	=>	INV	=>	0
0	0	L	=>		=>	1

(d)

Figure 9. Operation values of (c) {R, R}, (d) {H, L}.

The third type in Figure 9 is the "Hold" State (either "Hold 0" or "Hold 1"). As in Figure 9(e), two state bits in Present State (PS) are static "0" while two state bits in Next State (NS) are "1" and "0". This requires the MSR cell state in "Hold 0". Figure 9(f) is for "Hold 1" since current states have two static "1" and next states are "1" and "0". In summary, when two operation values are {R, L}, MSR Cell State is set to "Hold 0"; and {F, H} corresponds to MSR Cell State "Hold 1".

PS	NS	OP		MSR Cell		PS
0	1	R	=>	Hold 0	=>	0
0	0	L	=>		=>	0

(e)

PS	NS	OP		MSR Cell		PS
1	0	F	=>	Hold 1	=>	1
1	1	H	=>		=>	1

(f)

Figure 9. Operation values of (e) {R, L}, (f) {F, H}.

PS	NS	OP		MSR Cell		PS
0	1	R	=>	Fail	=>	1
1	1	H	=>		=>	0

(g)

PS	NS	OP		MSR Cell		PS
1	0	F	=>	Fail	=>	0
0	0	L	=>		=>	1

(h)

Figure 9. Operation values of (g) {R, H}, (h) {F, L}.

The forth type in Figure 9 is the "Fail" state, which means that the MSR cell can not satisfy the given circuit conditions. As in Figure 9(g), the two Present States are "1" and "0" and two Next State bits are static "1", which is not possible. Another example for "Fail" is in Figure 9(h). In summary, when two operation values are types of {R, H} or {F, L}, MSR Cell State is set to "Fail".

3.3 Test Pattern Generation Algorithm for Oscillation Test (OTPG Algorithm)

The test generation algorithm for oscillation testability is outlined below. The input of the algorithm is an FSM model. Each state transition is a four-tupple (x, p, n, y), which represents input vector, present state, next state, and output vector, respectively. Let the distance between two vectors $d(v_1, v_2)$ be the number of bit differences where the two vectors are different. For example, d(-00, 11-) = 1, where – indicates a don't-care bit. This $d(v_1, v_2)$ is also known as Hamming distance.

```
Algorithm: Oscillation Test Pattern Generation (OTPG)
Input: a set of state transition function $T$
Output: a set of test vectors
  for each ($t_i, t_j \in T$)
    if ($d(y_i, y_j) > 0$ && $d(x_i, x_j) > 0$) {
        $x \leftarrow x_i \cap x_j$;
        calculate MSR states from Operation Table;
        if (no "Fail" state)
           record valid state pairs $p_i$, $p_j$ with input $x$;
  }
```

For example, consider state pair (a, e) under X=0 in Figure 6. After MSR State Transition Algorithm, we get the vector of the MSR cell state[2..0]=[INV, Bypass, Bypass] in Figure 10.

		state	$b_2b_1b_0$	OP value		
1st	Present state	a	0 0 0	L	L	L
	Next state	a	0 0 0			
2nd	Present state	e	1 0 0	H	L	L
	Next state	e	1 0 0			
				INV	Bypass	Bypass

Figure 10. The MSR cell state for state pair (a, e).

4. Experimental Results

We have conducted experiments on LGSyn91 of MCNC benchmark circuits whose statistics are shown in Table I. In order to make the proposed method effective, we should have enough oscillation signals in the outputs. Therefore, circuits with very few outputs and output signal transitions are not included in the experiment.

In the symbolic states of the first 16 LGSynth91 benchmark circuits, the states are symbolic and NOVA [9] is used for state assignment. In the remaining 5 ISCAS89 circuits, the binary codes of the states are known. The proposed oscillation TPG (OTPG) algorithm is called to generate oscillation test vectors. The FSMs are then synthesized and the test vectors are evaluated for stuck-at test efficiency. The results are shown in Table II. Columns 2 to 4 give the results of the proposed method. The column under *#t (osc)* indicates the number of oscillation tests generated by our algorithm, while *TE* is the test efficiency achieved by this set of tests. The forth column (*#t* (*scan*)) gives the number of extra scan test vectors required to achieve 100% test efficiency. The last column (#stv) indicates the number of test vectors to achieve 100% test efficiency if only scan test is used. Our pure oscillation test (*#t(osc)*) provides average test efficiency of more than 90.11%, and 100% test efficiency can be achieved by adding extra scan test (*#t* (*scan*)). Under the same 100% fault coverage and test efficiency for stuck-at faults, our proposed method (*#t* (*osc*)+*#t* (*scan*)) requires almost same total/average numbers of test patterns compared to the pure scan test, and it outperforms the pure scan tests at the average ratio of 2:1 in the test cases.

5. Concluding Remarks

We present a novel oscillation test architecture for sequential circuits, in which at-speed testing is possible. As a result, delay related faults are detectable. We develop MSR cells for this architecture, and propose an efficient algorithm for oscillation test generation. The proposed method requires approximately the same amount of test vectors as scan tests to achieve 100% test efficiency for stuck-at faults, and it outperforms the pure scan tests at the average ratio of 2:1 in the test cases. In the future, we shall also consider state assignment method that makes 100% test efficiency possible with the oscillation test only.

References

[1] P. Nigh, W. Needhan, K. Butler, P. Maxwell, and R. Aitken, “An experimental study comparing the relative effectiveness of functional, scan, *IDDQ*, and delay-fault testing,” in *Proc. IEEE VLSI Test Symp.*, pp. 459-464, 1997.

[2] P. Maxwell, I. Hartanto, and L. Bentz, “Comparing functional and structural tests,” in *Proc. IEEE Int’l Test Conf.*, pp. 400-407, 2000.

[3] C.C. Liaw, S.Y. Su, and Y.K. Malaiya, “Test generation for delay faults using stuck-at-fault test set,” in *Proc. IEEE Int’l Test Conf.*, pp. 167-175, 1980.

[4] N. Tendolkar, R. Raina, R. Woltenberg, X. Lin, B. Swanson, and G. Aldrich, “Novel techniques for achieving high at-speed transition fault test coverage for Motorola’s microprocessors based on PowerPCTM instruction set architecture,” in *Proc. IEEE VLSI Test Symp.*, pp. 3-8, 2002.

[5] M. Kaneko and K. Sakaguchi, “Oscillation fault diagnosis for analog circuits based on boundary search with perturbation model,” in *Proc. IEEE Intnl Symp on Circuits and Systems*, pp93-96, 1994.

[6] K. Arabi and B. Kaminska, “Oscillation-based test strategy for analog and mixed-signal integrated circuits,” in *Proc. IEEE VLSI Test Symp.*, pp. 476-482, 1996.

[7] K. S.-M. Li, C.-L. Lee, C. Su, and J. E Chen, “Oscillation Ring Based Interconnect Test Scheme for SOC,” *Proc. IEEE ASPDAC*, pp. 184-187, 2005.

[8] S. Mitra, N. Seifert, M. Zhang, Q. Shi, K.S. Kim, “Robust system design with built-in soft-error resilience,” *IEEE Computer*, Volume 38, Issue 2, pp. 43-52, Feb. 2005.

[9] T. Villa and Sangiovanni-Vincentelli, “NOVA: State assignment of finite state machine of optimal two-level logic implementation,” *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. no. 9, pp. 905-924, 1990.

Table I. Statistics of benchmark circuits

Circuit	#input	#output	#states
bbsse	7	7	16
cse	7	7	16
dk14	3	5	7
dk15	3	5	4
dk16	2	3	27
dk17	2	3	8
dk27	1	2	7
dk512	1	3	15
lion	2	1	4
mc	3	5	4
planet	7	19	48
s1	8	6	20
sand	11	9	32
sse	7	7	16
styr	9	10	30
tbk	6	3	32
s27	4	1	6
s298	3	6	218
s386	7	7	13
s1488	8	19	48
s1494	8	19	48

Table II. Comparison of experimental results between Oscillation Test Pattern Generation (OTPG) and Pure Scan.

Circuit	**Proposed**			***#stv***
	#t (osc)	***TE* (%)**	***#t(scan)***	**(scan only)**
bbsse	28	83.87	21	52
cse	50	87.35	23	76
dk14	20	96.09	6	36
dk15	5	60.61	15	24
dk16	60	98.66	5	65
dk17	15	98.15	1	21
dk27	5	90.74	2	11
dk512	17	98.31	1	24
lion	4	90.00	3	8
mc	7	100.00	0	10
planet	120	97.08	20	128
s1	88	92.48	20	105
sand	152	99.60	3	140
sse	28	83.87	21	52
styr	154	98.35	9	157
tbk	87	57.81	119	189
s27	6	97.37	1	11
s298	42	98.87	7	30
s386	26	75.12	21	42
s1488	115	95.34	17	135
s1494	116	92.58	27	154
Average		**90.11**		