



On Using Twisted-Ring Counters for Test Set Embedding in BIST*

SHIVAKUMAR SWAMINATHAN

PowerPC Embedded Processor Development, IBM Corporation, Research Triangle Park, NC 27709, USA

s9s@us.ibm.com

KRISHNENDU CHAKRABARTY

*Department of Electrical and Computer Engineering, Duke University, 130 Hudson Hall, Box 90291,
Durham, NC 27708, USA*

krish@ee.duke.edu

Received September 26, 2000; Revised July 16, 2001

Editor: K.-T. Cheng

Abstract. We present a novel built-in self-test (BIST) architecture for high-performance circuits. The proposed approach is especially suitable for embedding precomputed test sets for core-based systems since it does not require a structural model of the circuit, either for fault simulation or for test generation. It utilizes a twisted-ring counter (TRC) for test-per-clock BIST and is appropriate for high-performance designs because it does not add any mapping logic to critical functional paths. Test patterns are generated on-chip by carefully reseeding the TRC. We show that a small number of seeds is adequate for generating test sequences that embed complete test sets for the ISCAS benchmark circuits.

Instead of being stored on-chip, the seed patterns can also be scanned in using a low-cost, slower tester. The seeds can be viewed as an encoded version of the test set that is stored in tester memory. This requires almost 10X less memory than compacted test sets obtained from ATPG programs. This allows us to effectively combine high-quality BIST with external testing using slow testers. As the cost of high-speed testers increases, methodologies that facilitate testing using slow testers become especially important. The proposed approach is a step in that direction.

Keywords: deterministic BIST, non-intrusive testing, reseeding, scalable BIST, test-per-clock

1. Introduction

A large number of test patterns must be applied to high-performance circuits in order to detect and diagnose realistic defects [11]. This can be achieved using a test-per-clock built-in self-test (BIST) architecture in which a test pattern is applied to the circuit under test (CUT) every clock cycle [4, 7, 17] and the test response is captured by an appropriate response monitoring circuit

[2, 3]; see Fig. 1(b). Unfortunately, most existing BIST methods rely on test-per-scan architectures that serially load test patterns into the scan chain [10, 13]. In such test-per-scan methods, as shown in Fig. 1(b), one test pattern is applied to an n -input CUT every $n + 1$ cycles. While test-per-clock BIST methods require less testing time, they introduce mapping logic between the input scan register and the CUT. The associated performance degradation makes these methods unsuitable for testing high-performance circuits. For example, the test-per-clock methods described in [7] and [17] require combinational mapping logic for transforming

*This research was supported in part by the National Science Foundation under grant no. CCR-9875324.

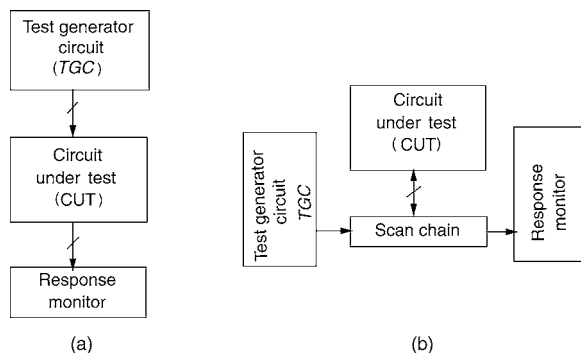


Fig. 1. Illustration of (a) test-per-clock and (b) test-per-scan BIST.

pseudorandom patterns generated by a linear-feedback shift-register (LFSR) to useful test patterns targeted at hard-to-detect faults.

A recently-proposed test-per-clock BIST architecture addresses the problem of applying patterns test-per-clock to high-performance circuits [4, 5]. It is based on reconfiguring the input scan register into a twisted ring counter (TRC), also known as a Johnson counter. The key idea is to employ reseeding to embed an entire precomputed deterministic test set T_D in a slightly longer test sequence that is applied to the CUT. The TRC-based test pattern generator can be designed by adding a multiplexer and an inverter to the serial input of the scan register feeding the CUT. Since no redesign of the CUT is necessary and no additional logic is added to the critical paths beyond that required for scan, this test architecture is especially suitable for high-performance circuits.

Another test set embedding scheme for scan-BIST based on a new type of test pattern generator, called a folding counter, has been recently proposed [9]. A folding counter can be designed using a Johnson counter with programmable feedback. This architecture can be easily extended to allow mixed-mode testing using pseudorandom patterns for detecting the easy faults. However, the testing time associated with this method can be very high since it is a test-per-scan approach. Moreover, this scheme employs test-width compression using input reduction, which can adversely impact the place-and-route of the scan flip-flops.

The TRC-based architecture of [4] offers a number of additional advantages. The test control logic is simple and can result in a lower overhead implementation than that required for LFSRs. In addition, the number of seed bits required is in many cases less than for previously-published (LFSR-based)

reseeding schemes. Finally, the TRC configuration is CUT-independent; it can be employed for testing multiple CUTs by simply changing seeds. In fact, this aspect of the TRC-based architecture can be especially appealing for BIST tool vendors who tend to prefer generic BIST circuits that do not have to be tailored to specific CUTs.

Despite the above advantages, there a number of ways in which the TRC-based test architecture of [4] can be improved. Since the seeds are stored on-chip, it is essential that a very small number of seeds must be sufficient for embedding T_D . Unfortunately, the number of seeds for the test architecture of [4] is sometimes too high. Furthermore, the *pattern efficiency* η in [4], defined as the number of test patterns applied to the CUT per clock cycle ($0 \leq \eta \leq 1$), is only 0.75. While this is significantly higher than the value of η for test-per-scan methods, even higher pattern efficiency is necessary for effective testing of today's high-performance circuits.

In this paper, we present a new BIST architecture based on the twisted-ring counter. For every seed that is serially loaded into the TRC, $O(n^2)$ test patterns are applied to an n -input CUT. In this way, a smaller number of seeds is now sufficient for embedding T_D . The seed count is also reduced from [4] since a more efficient algorithm is now used for seed selection. The low seed count translates to lower hardware overhead than in [4]. Furthermore, the pattern efficiency η is almost one for this test architecture; in fact, η asymptotically approaches one as n approaches infinity. Thus, not only do we achieve complete coverage of modeled faults, but we also increase the likelihood of detecting non-modeled faults by applying a large number of patterns to the CUT.

An important potential application of the proposed method lies in the on-chip generation of precomputed test patterns for embedded cores with boundary scan. Such cores usually come with functional or precomputed, deterministic test sets [18]. The TRC-based embedding approach can be used to design a pattern source for core-level BIST with acceptable area costs, especially since it does not require a gate-level model of the CUT. Hence the proposed approach is especially appealing when fault simulation and test generation for random-pattern-resistant faults are not feasible. This is in contrast to mixed-mode BIST methods that require fault simulation to determine random-testable faults and test generation to determine test cubes for random-resistant faults [9].

As in [4], the small number of seed patterns for the TRC can be stored on-chip or scanned in using a low-cost, slower tester. The patterns derived from the seeds can then be applied to the CUT. The pattern efficiency asymptotically approaches one, even if the seeds are scanned in externally from the tester. This allows us to effectively combine high-quality BIST with external testing using slow testers. This is especially important since the International Technology Roadmap for Semiconductors (ITRS) predicts that the cost of high-speed testers will exceed \$20 million by 2010 [11]. While IC speeds have improved at the rate 30% per year, tester accuracy has improved at an annual rate of only 12%. Hence, test methods that can be used with low-cost, slower testers will become increasingly important in the near future [1, 14].

Yet another advantage of the proposed method lies in reduced tester memory. The seeds can be viewed as an encoded version of the test set that is stored in tester memory. This requires almost 10X less memory than compacted test sets derived from automatic test pattern generation (ATPG) programs. This demonstrates that if the seeds are stored in tester memory, test set compaction may not always be necessary.

The organization of this paper is as follows. Section 2 introduces the new TRC-based test architecture. Section 3 addresses the seed selection problem and describes our seed selection algorithm. Finally, Section 4 presents experimental results, comparisons with [4] and [10], and presents some design guidelines based on the interpretation of experimental results.

2. Test Architecture

In this section, we first review the operation of a twisted-ring counter (TRC) and then describe the new TRC-based BIST architecture. An n -bit ring counter is a group of n flip-flops F_1, F_2, \dots, F_n connected as a shift register, with the output of F_n fed back to the input of F_1 . It behaves as a counter with up to n distinct states, depending on the initial value (seed). The TRC is a ring counter with an inverter added between the output of F_n and the input of F_1 . Depending on the seed, an n -bit TRC behaves as a counter with up to $2n$ distinct states.

The operation of a TRC as a BIST pattern generator is sketched in Fig. 2. The ROM for storing seeds can be eliminated if an external tester is used to scan in the seed patterns. The CUT's n -bit input register is configured into a TRC of length n during testing. The

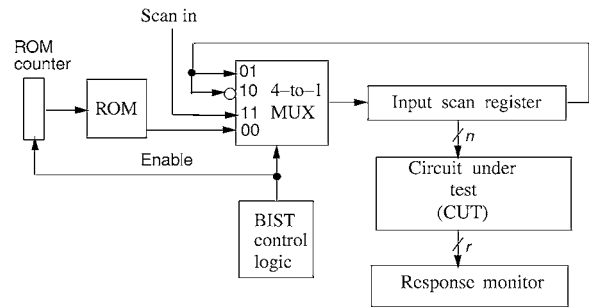


Fig. 2. The TRC-based BIST architecture of [4].

pattern generator operates in two modes, namely the “shift” and “twist” modes, which correspond to an n -bit ring counter and TRC, respectively. In [4], the TRC was clocked for $3n$ cycles for every seed; see Fig. 3(a). In this paper, we exploit the fact that a much larger number of patterns can be generated from a given seed by clocking the TRC for $2n^2 + n$ cycles. This is shown in Fig. 3(b) where a total of s seeds are used for test application.

The test control circuit is derived from a simple finite-state machine (FSM) with three states—Load, Twist and Shift. The state diagram for the Moore-type FSM is shown in Fig. 4. (The Twist state is depicted in the state diagram as two substates—Twist1 and Twist2.) Two k -bit counters, $k = \lceil \log_2 n \rceil$, with enable inputs are required to generate the signals “Twist Enable” (TE) and “Shift Enable” (SE) that feed the test control logic. One of these counters is required in any autonomous scan-based BIST scheme [13]. The control circuit generates a “Shift-Counter Enable” (SCE) signal which enables the shift counter.

We now describe the operation of the finite-state machine.

1. The FSM is initially in the Load state (encoded as the 00 binary pattern). The seed pattern is serially loaded into the n -bit scan register in n clock cycles.
2. At the end of the Load operation, TE goes high and the FSM goes into the Twist state (encoded as two substates corresponding to the binary patterns 01 and 10, respectively). The content of the input register undergoes $2n$ twists in $2n$ clock cycles. After $2n$ cycles, scan register contains the seed pattern.
3. After $2n$ twists are completed, SCE goes high, and the FSM goes into the Shift state (encoded as the 11 binary pattern). It remains in this state for only one clock cycle. A 1-bit shift is performed on the content of the input register.

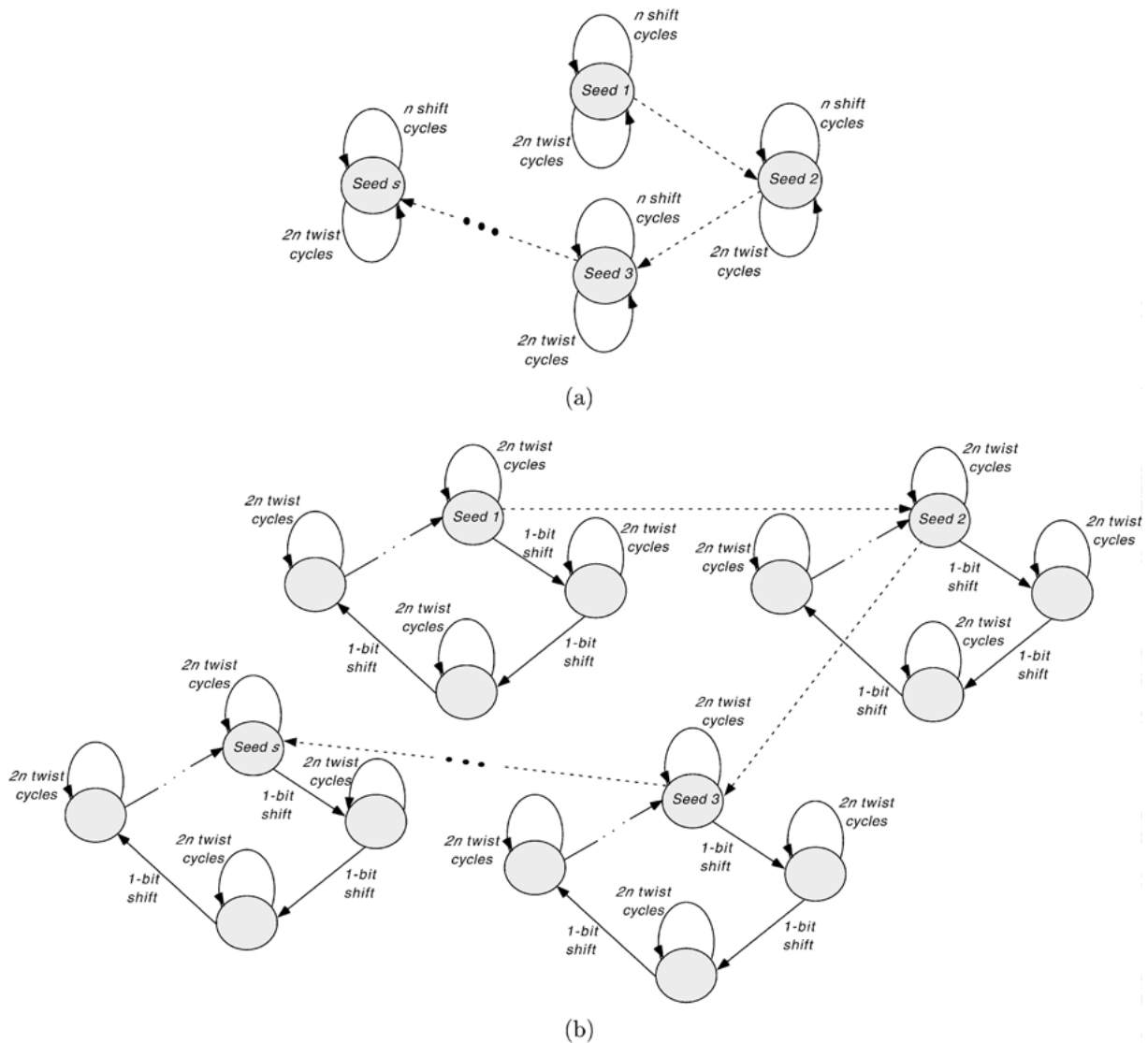


Fig. 3. Pattern generation using a TRC: (a) $3n$ cycles per seed [4]; (b) the proposed method based on $2n^2 + n$ cycles per seed.

4. The FSM now re-enters the Twist state, and remains in this state for $2n$ cycles. This process repeats until n shift operations are completed, at which point SE goes high and a new seed is loaded into the input register.

The 3-state FSM can be implemented using two flip-flops and a small amount of combinational logic. An implementation of the control logic is shown in Fig. 5. In order to verify its functionality, we used the Mentor Graphics DA tool for schematic entry and Quicksim for logic simulation. The test control logic is

CUT-independent and scalable. Only the counters have to be redesigned to account for a different value of k when n changes.

It can be easily seen that upto $2n^2 + n$ patterns are applied to the CUT for every seed that is scanned into the input register. This is considerably larger than $3n$ patterns that are generated from a seed in [4]. Even though the test application time is increased, the larger number of patterns applied to the CUT increases the likelihood of detecting non-modeled faults and improves overall test quality.

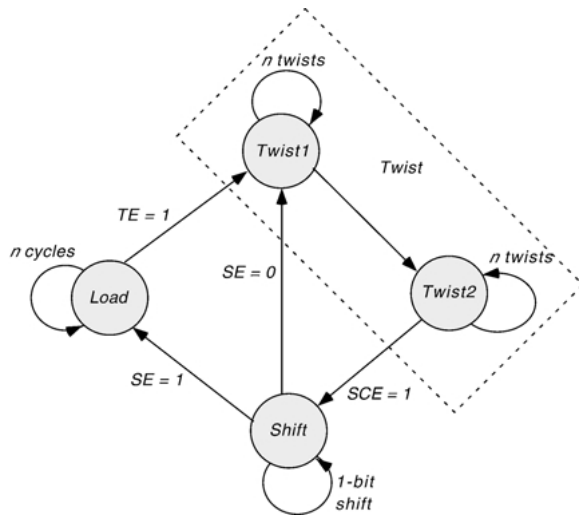


Fig. 4. The finite-state machine for the test control circuit.

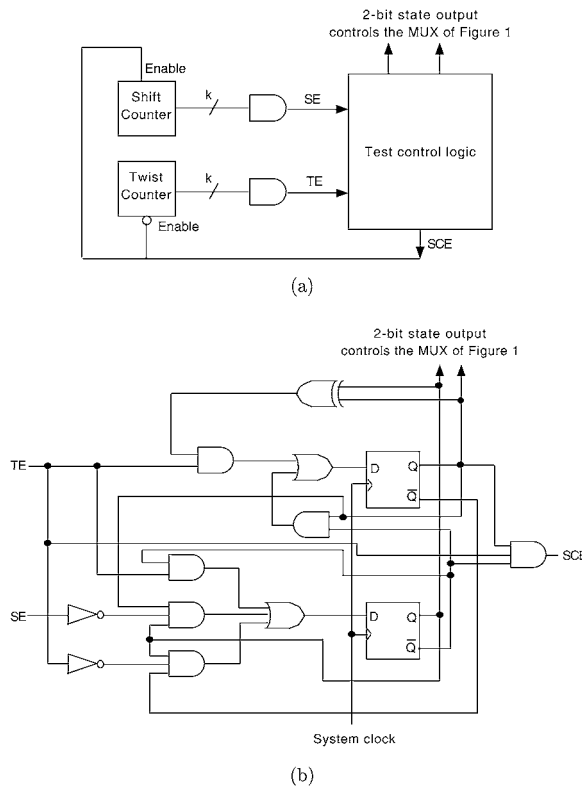


Fig. 5. The test control logic for the proposed BIST architecture: (a) block diagram; (b) detailed schematic diagram.

Note that it is possible that a few of the patterns generated by the TRC may be repetitions of patterns already applied to the CUT. However, this seems to occur rarely. A general characterization of TRC sequence

cycle length for arbitrarily chosen seeds remains an interesting open problem.

We next evaluate the pattern efficiency of the proposed test architecture. A total of ns cycles are required for loading s seeds into the scan register. Test patterns are applied test-per-clock for $(2n^2 + n)s$ cycles when the FSM of Fig. 4 is in the Shift and Twist states. The pattern efficiency η is therefore given by

$$\eta = \frac{(2n^2 + n)s}{(2n^2 + n)s + sn} = \frac{1}{1 + \frac{1}{2n+1}}$$

It follows therefore that η is not only high, but it also rapidly approaches 1 (ideal case) as n increases (Fig. 6).

The seed patterns can also be scanned in using a low-cost, relatively slow tester. The patterns derived from the seeds can then be applied to the CUT. This allows us to effectively combine highquality BIST with external testing using slow testers. We now show that extremely high pattern efficiency is obtained even if the seeds are stored on a slow, external tester. Let the tester frequency be f_{ext} and let the scan clock frequency be f_s . The pattern efficiency η is then given by

$$\eta = \frac{(2n^2 + n)s/f_s}{(2n^2 + n)s/f_s + sn/f_{ext}} = \frac{1}{1 + \frac{f_s}{(2n+1)f_{ext}}}$$

Fig. 7 shows that high pattern efficiency can easily be achieved, even with slow testers. Once again, near-ideal pattern efficiency is achieved since $\eta \rightarrow 1$ as $n \rightarrow \infty$ for any given f_s/f_{ext} ratio.

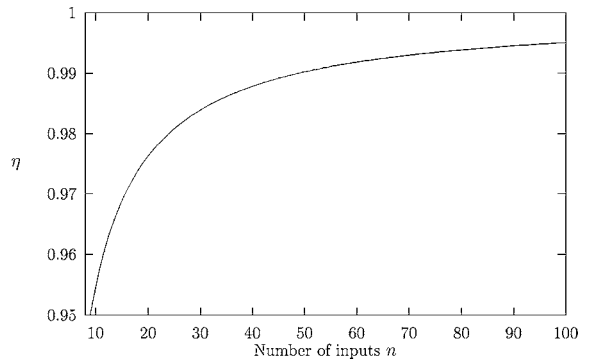


Fig. 6. Variation of pattern efficiency with the number of inputs n .

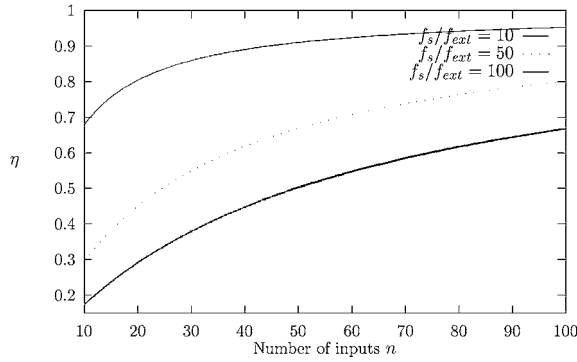


Fig. 7. Variation of pattern efficiency with the number of inputs n for various f_s/f_{ext} ratios.

If the seeds are stored in an on-chip ROM, then the counter addressing the ROM is clocked at the scan rate while the scan chain is loaded with a seed pattern. On the other hand, if the seeds are stored in an external tester, they must be scanned in at the frequency of the external tester. This is usually lower than the on-chip scan frequency. Once the seed is scanned in, the TRC runs independently of the tester, hence no explicit synchronization is required between the TRC clock and the external tester clock.

3. Seed Selection

In this section, we describe an algorithm for selecting a small set of seeds that generates all the test patterns in T_D . We first review some terminology and then present a lower bound on the number of seeds required for embedding T_D .

The *transition count* of a fully-specified pattern t_j is the sum of the number of 1-to-0 and 0-to-1 transitions in t_j . If t_j is defined as the vector $b_1b_2 \dots b_n$ then the transition count $TC(t_j) = \sum_{i=1}^{n-1} b_i \oplus b_{i+1}$. A pattern t_j is *compatible* with pattern t_k if there is no bit position in t_j and t_k that are specified and different. For example, 1011X is compatible with X01X0 but is not compatible with 11X1X because of the conflict in the second bit position. Finally, a pattern t_k lies on the shift cycle of pattern t_j if it can be obtained from t_j by performing at most n consecutive shift operations. Similarly, t_k lies on the twist cycle of t_j if it can be obtained from t_j by performing at most $2n$ consecutive twist operations.

Lemma 1. *If a pattern t_k lies either on the shift cycle or on the twist cycle of another pattern t_j , then $|TC(t_j) - TC(t_k)| \leq 1$.*

Proof: We consider the following three cases:

Case 1. The pattern t_k lies on the shift cycle of t_j . Suppose $t_j = b_1b_2 \dots b_n$, and t_k is obtained from t_j after $i \leq n$ shift operations. This implies that $t_k = b_{n-i+1}b_{n-i+2} \dots b_nb_1 \dots b_{n-i}$. Now, $TC(t_j) = (b_1 \oplus b_2) + (b_2 \oplus b_3) + \dots + (b_{n-1} \oplus b_n)$ and $TC(t_k) = (b_{n-i+1} \oplus b_{n-i+2}) + \dots + (b_1 \oplus b_n) + \dots + (b_{n-i-1} \oplus b_{n-i})$. Therefore, $|TC(t_j) - TC(t_k)| = |(b_{n-i} \oplus b_{n-i+1}) - (b_n \oplus b_1)|$. This difference can be at most 1.

Case 2. The pattern t_k lies on the twist cycle of t_j . Suppose $t_j = b_1b_2 \dots b_n$, and t_k is obtained from t_j after $i \leq n$ twist operations. This implies that $t_k = \bar{b}_{n-i+1}\bar{b}_{n-i+2} \dots \bar{b}_nb_1b_2 \dots b_{n-i}$. Therefore, $|TC(t_j) - TC(t_k)| = |(b_{n-i} \oplus b_{n-i+1}) - (\bar{b}_n \oplus b_1)|$, and once again, this difference can be at most 1.

Case 3. The pattern t_k lies on the twist cycle of t_j . Suppose $t_j = b_1b_2 \dots b_n$, and t_k can be obtained from t_j after $n < i \leq 2n$ twist operations. Let $i = l \text{ mod } n$. Now, $t_j = b_{n-l+1}b_{n-l+2} \dots b_nb_1\bar{b}_2 \dots \bar{b}_{n-1}$. Therefore, $|TC(t_j) - TC(t_k)| = |(b_{n-l} \oplus b_{n-l+1}) - (b_n \oplus \bar{b}_1)|$, and once again, this difference can be at most 1. \square

Lemma 1 implies that if at most $3n$ patterns are generated from a seed as in Fig. 3(a), then two test patterns t_j and t_k can be derived from the same seed only if $|TC(t_j) - TC(t_k)| \leq 1$. This is however only a necessary condition; in fact, we can easily identify pairs of patterns whose transition counts differ by at most 1, yet they cannot be derived from the same seed.

A lower bound on the number of seeds can now be obtained by constructing a graph $G(V, E)$, where the set of vertices V denotes the test patterns in T_D and $(t_j, t_k) \in E$ if $|TC(t_j) - TC(t_k)| > 1$. A lower bound on the number of seeds is equal to the chromatic number of G , i.e. the number of colors required for a proper coloring of G .

In order to determine a lower bound for test sets that contain partially-specified patterns (test cubes), we define the maximum transition count $TC_{\max}(t_j)$ and minimum transition count $TC_{\min}(t_j)$ for test patterns t_j as follows. The maximum transition count $TC_{\max}(t_j) = TC(t_{j,1})$ where $t_{j,1}$ is obtained from t_j by replacing the subsequences containing don't-cares in t_j in the following manner:

- (i) Replace $1X \dots X0$ by $1010 \dots 0$
- (ii) Replace $0X \dots X1$ by $0101 \dots 1$

- (iii) Replace $1X\dots X1$ by $1010\dots 1$, and
- (iv) Replace $0X\dots X0$ by $0101\dots 0$.

Similarly, $TC_{\min}(t_j) = TC(t_{j,2})$ where $t_{j,2}$ is obtained from t_j by replacing the subsequences containing don't-cares in t_j as follows:

- (i) Replace $1X\dots X0$ by $111\dots 0$
- (ii) Replace $0X\dots X1$ by $000\dots 1$
- (iii) Replace $1X\dots X1$ by $111\dots 1$, and
- (iv) Replace $0X\dots X0$ by $000\dots 0$.

It can now be easily shown that if two partially-specified patterns t_j and t_k are derived from the same seed, then either $TC_{\min}(t_j) - TC_{\max}(t_k) \leq 1$ or $TC_{\min}(t_k) - TC_{\max}(t_j) \leq 1$.

Recall that in the proposed BIST architecture, we perform $2n$ shift operations on each of n patterns that are obtained by carrying out shift operations on a seed; see Fig. 3(b). The following theorem, which is based on Lemma 1, provides a necessary condition for two patterns to be derived from the same seed during $2n^2 + n$ clock cycles.

Theorem 1. *If patterns t_j and t_k are derived from a common seed s during the $2n^2 + n$ cycles in which s is used for pattern generation, then $|TC(t_j) - TC(t_k)| \leq 3$.*

Proof: We prove the theorem by considering the following cases:

Case 1. Suppose t_j is either the seed s or is obtained by performing shift operations on s . Suppose t_k is also obtained from s by carrying out shift operations. From Lemma 1, it follows that since t_j and t_k lie on the shift cycle of s , $|TC(t_j) - TC(t_k)| \leq 1$.

Case 2. Suppose t_j is the seed s and t_j lies on the twist cycle of s . Once again, from Lemma 1, it follows that $|TC(t_j) - TC(t_k)| \leq 1$. This also holds if t_j lies on the shift cycle of s and t_k lies on the twist cycle of t_j .

Case 3. Suppose t_j lies on the shift cycle of s and t_k lies on the twist cycle of another pattern t_i on the shift cycle of s . By applying Lemma 1, we obtain $|TC(t_j) - TC(t_i)| \leq 1$ and $|TC(t_k) - TC(t_i)| \leq 1$. Therefore, $|TC(t_j) - TC(t_k)| \leq 2$.

Case 4. Let t_u and t_v lie on the shift cycle of s and let t_j (t_k) lie on the twist cycle of t_u (t_v). Since

$$\begin{aligned} |TC(t_u) - TC(t_v)| &\leq 1 \\ |TC(t_u) - TC(t_j)| &\leq 1 \\ |TC(t_k) - TC(t_v)| &\leq 1 \end{aligned}$$

it follows that $|TC(t_j) - TC(t_k)| \leq 3$. This completes the proof. \square

A lower bound on the number of seeds can now be obtained using graph coloring. For partially-specified test sets, we can once again determine the maximum and maximum transition counts by appropriately assigning binary values to the don't-care bits.

We now address the problem of selecting the seeds to be used for generating test patterns using a TRC. Note that the problem of determining the smallest set of seeds is NP-hard since it can be formulated in terms of clique partitioning. Therefore, we present a heuristic algorithm for determining the seeds in reasonable time. The seed selection procedure is described in pseudocode form in Fig. 8.

The heuristic procedure works as follows. An arbitrary ordering is first imposed on the test patterns in T_D . The first (partially-specified) test pattern in the test set T_D is then chosen as the starting seed s . This seed undergoes $2n$ twists. If a pattern on the twist cycle is compatible with a test pattern in T_D , the don't care bits in the generated pattern (and therefore the don't-care bits in s) are appropriately assigned binary values and the corresponding test pattern is marked as "covered." (The objective is to keep as many don't-care bits unassigned as possible since more don't-cares increase the likelihood of covering patterns from T_D .) This is followed by a 1-bit shift operation on s , and the process is repeated until all n shift operations are carried out. At the end of n shift operations (complete shift cycle of s), if there are test patterns that are not yet covered, the first "unmarked" (partially-specified) test pattern is chosen as the next seed. The procedure terminates when all patterns in the test set are covered. The worst-case complexity of this procedure is $O(m^2n^2)$.

For example, consider the c17 ISCAS benchmark circuit with the test set T_D shown in Fig. 9. We choose the first test pattern t_1 in the test set as our starting seed. The pattern obtained by carrying out a 1-bit twist on t_1 is compatible with t_9 . For this example, the seed becomes fully-specified at this point. In general however, several matches will be obtained before all the

```

Procedure SelectSeed()
  /* select a set of seeds to generate test patterns in  $T_D$  */
  begin
     $k := 1$ ; /*seed index */
    shift_count := 0;
     $S := \phi$ ; /*initialize seed set */
     $s_k := t_1$ ; /* first test pattern chosen as first seed */
    while(all_test_patterns_not_marked)
      while(shift_count  $\neq n$ )
        for(number_of_twists = 1 to  $2n$ )
          begin
            twist( $s_k$ );
            for( $j := 1$  to  $m$ )
              if compatible( $s_k, t_j$ ) and mark( $t_j$ ) = 0
                /* Check whether  $s_k$  and  $t_j$  are compatible and  $t_j$  not yet covered */
                break;
              end for;
             $s_k := \text{assign}(t_j)$ ;
            mark( $t_j$ ) := 1;
            /* assign don't cares in  $s_k$  appropriately and mark  $t_j$  as covered */
          end for;
          one_bit_shift( $s_k$ ); Increment shift_count;
          /* Carry out a 1-bit shift on  $s_k$  */
          for( $j := 1$  to  $m$ )
            if compatible( $s_k, t_j$ ) and mark( $t_j$ ) = 0
              break;
            end for;
             $s_k := \text{assign}(t_j)$ ;
            mark( $t_j$ ) := 1;
          end while;
           $S := S \cup s_k$ ;
          Increment  $k$ ;
          /* add  $s_k$  to seed set */
          for( $j := 1$  to  $m$ )
            if (mark( $t_j$ ) = 0)
               $s_k := t_j$ ;
              break;
            /* pick next seed  $s_k$  */
            end if;
          end for;
        end while;
      return( $S$ );
    end Procedure;

```

Fig. 8. The seed selection procedure.

don't-care bits of the seed are assigned to 1s and 0s. Fig. 9 illustrates the matches obtained as t_1 performs the specified sequence of shifts and twists. All the patterns in T_D are covered after 31 cycles, and the pattern 01100 is selected as the seed. The patterns from the generated sequence that cover test patterns from T_D are shown shaded in the figure.

The above seed selection procedure carefully assigns binary values to the don't-care bits in candidate seed patterns. This is in contrast to the simple procedure of [4] in which the don't-cares were randomly assigned to 1s and 0s. As a result, compared to [4], the number

of seeds is considerably reduced. The reduction in the seed count can also be partly attributed to the larger number of patterns generated by the TRC.

4. Experimental Results

In this section, we present experimental results on the number of seeds required by the proposed BIST architecture for the ISCAS benchmark circuits. We only consider circuits that are not completely tested with 10,000 pseudorandom patterns. We used

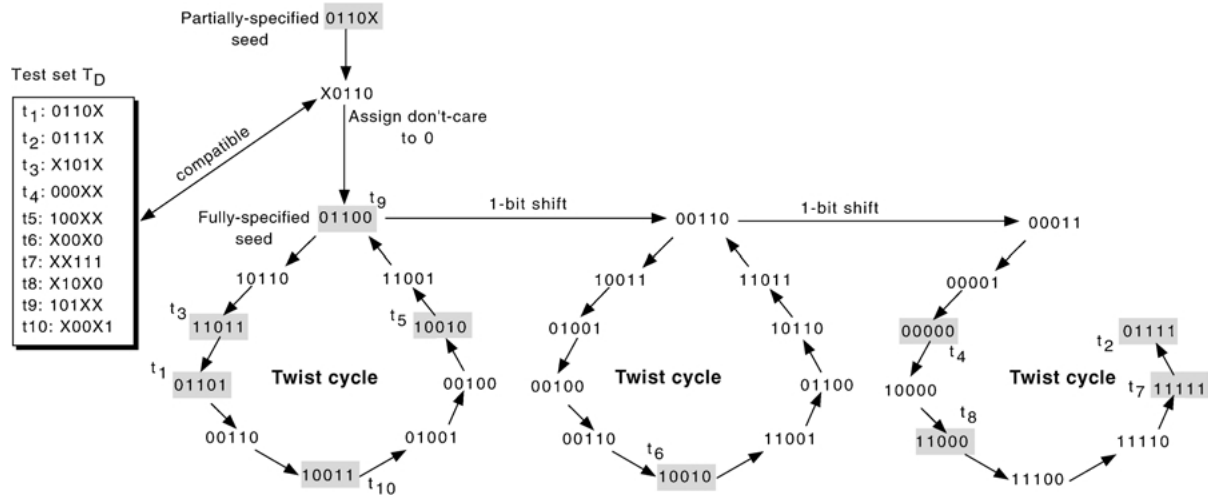


Fig. 9. Illustration of the seed selection algorithm for the c17 benchmark circuit.

partially-specified test sets obtained by instrumenting the Atlanta ATPG program [15], as well partially-specified test sets obtained from the Mintest ATPG program [8]. All these test sets provide 100% coverage of detectable single stuck-at faults. We carried

out our experiments on an Ultra-10 Sun Workstation with a 333 MHz processor and 128 MB of DRAM.

The seeds may either be stored in a ROM as in [10], or generated using combinational logic as in [7, 17]. Note that [10] is test-per-scan techniques, while

Table 1. Experimental results on the number of seeds required for the ISCAS 89 circuits. (For our experiments, we used test sets without dynamic compaction from Mintest.)

ISCAS 89 circuit	Number of primary inputs (n)	Number of test patterns (m)	Number of seeds (s)	Number of bits (N_1)	Number of bits in [10], (N_2)	(N_1/N_2)	Number of bits in [4], (N_3)	(N_1/N_3)	Number of bits in [9], (N_4)
s349	24	126	3	72	98	0.73	96	0.75	N/A
s386	13	120	9	117	466	0.25	69	1.69	N/A
s420	35	161	9	315	689	0.46	455	0.69	588
s510	25	130	4	100	218	0.46	125	0.80	N/A
s526	24	207	7	168	568	0.29	264	0.64	N/A
s641	54	231	6	324	672	0.48	594	0.55	288
s713	54	220	8	432	686	0.63	648	0.67	270
s820	23	253	11	253	949	0.27	414	0.61	N/A
s832	23	255	11	253	933	0.27	437	0.58	N/A
s838	67	319	31	2077	2013	1.03	2144	0.97	493
s953	45	256	8	360	796	0.45	855	0.42	1083
s1196	32	346	15	480	1578	0.30	1248	0.38	1178
s1238	32	360	25	800	1614	0.50	960	0.90	1850
s5378	214	1458	18	3852	3973	0.97	N/A	N/A	3180
s9234	247	1928	50	12350	10426	1.18	N/A	N/A	8342
s13207	700	3237	2	1400	7684	0.18	N/A	N/A	3471
s15850	611	3920	11	6721	9302	0.72	N/A	N/A	7498
s38417	1664	10771	19	31616	37622	0.84	N/A	N/A	35500
s38584	1464	13468	22	32208	13062	2.47	N/A	N/A	7298

Table 2. Experimental results on the number of seeds for the (a) Mintest test sets with dynamic compaction, (b) Atalanta test sets.

ISCAS 89 circuit	Number of primary inputs (n)	Number of test patterns (m)	Number of seeds (s)
s349	24	15	7
s386	13	64	9
s400	24	28	8
s420	35	51	16
s510	25	61	3
s526	24	55	16
s641	54	26	13
s713	54	26	14
s820	23	104	15
s832	23	109	14
s838	67	86	25
s953	45	88	10
s1196	32	139	21
s1238	2	152	24
s5378	214	111	37
s9234	247	159	70
s13207	700	236	32
s15850	611	126	43
s349	24	85	3
s386	13	102	6
s400	24	96	2
s420	35	129	8
s510	25	103	5
s526	24	160	7
s641	54	200	9
s713	54	192	8
s820	23	195	9
s838	67	257	15
s953	45	227	9
s1196	32	291	17
s1238	32	164	16
s9234	247	1521	33

[7, 17] are test-per-clock methods. We present our results in Tables 1 and 2. In addition, Table 1 provides a comparison with the results presented in [4, 9, 10]. Note however that the technique described in [10]: (a) implements a mixed-mode approach, (b) is targeted at scan-based-BIST using LFSRs, and (c) has more complex feedback and control logic. We also compare our

results with the test-per-clock TRC reseeding approach presented in [4].

The number of bits that need to be stored for the proposed method (N_1) is lower than in [10] (N_2) for most of the full-scan versions of the ISCAS 89 circuits, even though the total number of test patterns embedded is higher than the number of patterns embedded in [10]. For the benchmark circuits considered the proposed method requires on average 31% less bits than in [10] and 26% less bits than in [4] for storing seeds. The proposed method also requires less bits than [9] for storing seeds for a number of circuits. We do not report results for the ISCAS 85 circuits since our method does not improve on the results reported in [10]. Our test sets for the ISCAS 85 circuits have relatively few don't cares, therefore more seeds are needed for deterministic testing. Except for the last four circuits in Table 1, the CPU time for determining the seeds is less than 6 seconds. For the four larger circuits, the CPU time is higher (of the order of a few hours); however, this is only a one-time cost incurred during design. We are nevertheless improving the efficiency of the seed selection procedure by pruning the search space using our lower bound results.

The number of seeds is also considerably less than in the TRC-based reseeding method of [4]. The trade off involved here is a corresponding increase in the test application time. This is described in more detail in Table 3. The results show that even though on average, the test time increases by over 10X compared to [4], it is still quite low in a BIST environment, where frequencies of 500 MHz are becoming common. This can however, potentially lead to high power consumption in system-on-a-chip, where BIST may be applied concurrently to multiple cores. One way to address this problem is to use SOC test scheduling methods, such as the approach presented in [12]. In a typical SOC with multiple cores, a combination of BIST and external testing is used. In order to limit power consumption during testing, we use the fact that since external patterns are applied at a slower tester frequency, BIST tests for any given core can be overlapped with external testing of another core.

As discussed in Section 2, the proposed test architecture allows BIST to be combined with external testing using slower testers. The seeds can be stored in the tester memory and serially loaded into the TRC. The patterns derived from the seeds are then applied to the CUT. This obviates the need for storing seeds on-chip and therefore reduces BIST overhead substantially; yet

Table 3. Comparison of the testing time for proposed method compared to [4] for Atalanta test sets.

ISCAS 89 circuit	Number of inputs(n)	Number of clock cycles ($2n^2s + ns + ns$)	Number of clock cycles [4] ($4ns$)	Test time ^a (μs)	Test time ^a [4] (μs)
s349	24	3600	384	7.20	0.77
s386	13	2184	156	4.37	0.31
s420	24	20160	1820	43.20	3.64
s510	25	6500	500	13.0	1.00
s526	24	8400	1056	16.8	2.12
s641	54	35640	2376	71.28	4.752
s713	54	47520	2592	95.04	5.184
s820	23	9936	1656	19.872	3.312
s838	67	149343	8576	298.69	17.152
s953	45	37260	3420	74.52	6.84
s1196	32	35904	4992	71.81	9.984
s1238	32	33792	3840	67.58	7.68
s9234	247	4.04×10^6	N/A	8.09 ms	N/A

^aClock frequency is assumed to be 500 MHz.

the increase in testing time (as shown in Table 4) is in most cases insignificant. Even when this increase exceeds 10%, the overall testing time is still less than 1 ms.

We found from our experiments that the fraction of don't-cares in the test set T_D has a major impact on the number of seeds that are required for embedding. In order to investigate this issue further, we define $\alpha =$

ϕ/mn as the fraction of don't-care bits in T_D (ϕ is the number of don't-cares) and $\beta = s/m$ as the ratio of the number of seeds to the size of the test set. We observed a strong correlation between these two parameters—in general, β tends to decrease with an increase in α . For the ISCAS 85 circuits, the test sets contain fewer don't-cares, hence test set embedding using a TRC is less effective in these cases. However, for the

Table 4. Test application times for the benchmark circuits with Atalanta test sets when external testing is combined with BIST.

ISCAS 89 circuit	Number of BIST cycles ($2n^2s + ns$)	Number of external test cycles (ns)	BIST time ^a (μs)	External test ^b (μs)	Total test time t (μs)	Percentage increase in t
s349	3528	72	7.06	1.44	8.50	18
s386	2106	78	4.21	1.56	5.77	32.04
s420	19880	280	39.76	5.6	45.36	5.0
s510	6375	125	12.75	2.5	15.25	17.3
s526	8232	168	16.46	3.36	19.82	17.80
s641	35316	324	70.63	6.48	77.11	8.18
s713	47088	432	94.18	8.64	102.82	8.19
s820	9729	207	19.46	4.14	23.60	18.76
s838	135675	13668	271.35	273.36	544.7	82.37
s1196	35360	544	70.62	10.88	81.6	13.63
s1238	33280	512	66.56	10.24	76.80	13.64
s9234	6.11×10^6	12350	12.23 ms	247	12.48 ms	1.88

^aBIST frequency: 500 MHz.

^bTester frequency: 50 MHz.

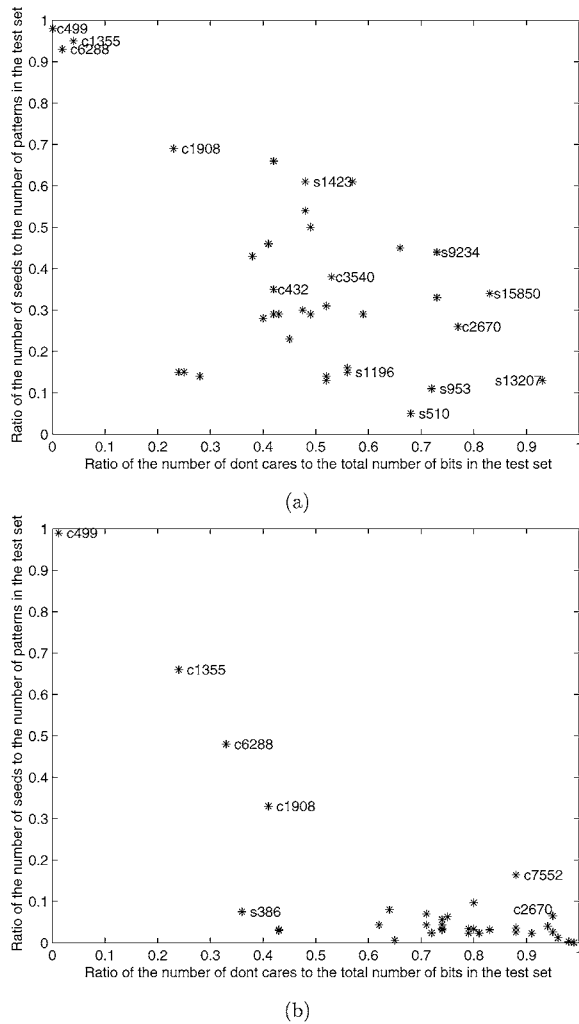


Fig. 10. Variation of α with β for the Mintest test sets (a) with dynamic compaction (b) without dynamic compaction.

ISCAS 89 circuits, α is significantly higher, hence these circuits tend to require fewer seeds. Fig. 10 presents scatter plots illustrating this observation.

This empirical observation can be used to evaluate the suitability of a test set for embedding. It appears that for test sets for which α is higher than a certain threshold, TRC-based test set embedding should not be attempted. We are currently investigating this issue further to determine the threshold value and to quantify the relationship between α and β .

Finally, we draw the following interesting conclusion from the experimental results on the number of seeds required for test set embedding. If the seeds are

stored in tester memory, then our results demonstrate that test set compaction may not always be necessary during automatic test pattern generation (ATPG). The seeds can be viewed as an efficient encoding of the test set obtained without compaction. In all the cases that we consider, the number of seeds is much less than the size of the smallest test set known for these circuits. In Table 5, we compare the amount of test data obtained using ATPG compaction to the amount of test data obtained by generating seeds for the test sets obtained without dynamic compaction. Almost 10X reduction in memory is achieved if only the seeds are stored in tester memory.

5. Conclusion

We have presented a new architecture for test-per-clock BIST based on the twisted-ring counter (TRC). The proposed approach is especially suitable for high-performance designs since it does not add any mapping logic to critical functional paths. It is also useful for core-level BIST since it does not require access to a gate-level model of the CUT.

Test patterns are generated on-chip by carefully re-seeding the TRC. For every seed that is serially loaded into the TRC, $O(n^2)$ test patterns are applied to an n -input CUT. We have shown that a small number of seeds is adequate for generating test sequences that embed complete test sets for the ISCAS benchmark circuits. We have also shown that the pattern efficiency η is almost one for this test architecture; in fact, η asymptotically approaches one as n increases. Thus, not only do we achieve complete coverage of modeled faults, but we also increase the likelihood of detecting non-modeled faults by applying a large number of patterns to the CUT. In future work, we expect to show that modeled faults are detected multiple times by patterns in TRC sequences; such n -detect tests are known to be effective for detecting defects that are not modeled by classical fault models [16].

A limitation of the proposed method is that it cannot be used to embed test sequences for sequential circuits, in which the ordering of the patterns must be preserved exactly. Nevertheless, this technique can be easily extended to embed two-pattern tests for delay faults. One way to do this is to use an enhanced-scan architecture [6], and view the concatenation of the two patterns in a vector pair as a single composite pattern. For a circuit with n primary inputs, we now have a $2n$ -bit TRC

Table 5. Comparison between TRC encoding and test set compaction during ATPG.

ISCAS 89 circuit	TRC Reseeding		ATPG compaction		Percentage reduction in tester memory
	Number of seeds	Size of encoded test set (bits)	Number of patterns (compacted)	Size of test set (bits)	
s349	3	72	23	552	86.96
s386	9	117	63	819	85.71
s420	9	315	43	1505	79.07
s510	4	100	54	1350	92.59
s526	7	168	49	1176	85.71
s641	6	324	21	1134	71.43
s713	8	432	21	1134	61.90
s820	11	253	93	2139	88.17
s832	11	253	94	2162	88.29
s838	31	2077	75	5025	58.67
s953	8	360	76	3420	89.47
s1196	15	480	113	3616	86.73
s1238	25	800	121	3872	79.34
s5378	18	3852	97	20758	81.44
s9234	50	12350	105	25935	52.38
s13207	2	1400	234	163800	99.15
s15850	11	6721	95	58045	88.42
s38417	19	31616	68	113152	72.06
s38584	22	32208	110	161040	80.0

and TRC sequences can be used to embed composite patterns.

Instead of being stored on-chip, the seed patterns can also be scanned in using a low-cost, slower tester. The seeds can be viewed as an encoding of the test set that is stored in tester memory. Our experimental results demonstrate that test set compaction may not always be necessary during automatic test pattern generation. In all the cases that we consider, the number of seeds is much less (10X) than the size of the smallest test set known for these circuits. During testing, the patterns derived from the seeds can be applied test-per-clock to the CUT. This allows us to effectively combine high-quality BIST with external testing using slow testers. We have shown that near-ideal pattern efficiency is achieved even if the seeds are stored in tester memory; furthermore, the increase in testing time is insignificant. As the cost of high-speed testers increases, test methodologies that facilitate testing using slow testers become especially important. The proposed approach presents a step in that direction.

References

1. V.D. Agrawal and T.J. Chakraborty, "High-Performance Circuit Testing using Slow-Speed Testers," in *Proc. International Test Conference*, 1995, pp. 302–310.
2. P. Bardell, W. McAnney, and J. Savir, *Built-in Test for VLSI. Pseudorandom Techniques*, New York, NY: John Wiley, 1987.
3. K. Chakrabarty, B.T. Murray, and J.P. Hayes, "Optimal Zero-Aliasing Space Compaction of Test Responses," *IEEE Transactions on Computers*, Vol. 47, pp. 1171–1187, November 1998.
4. K. Chakrabarty, B.T. Murray, and V. Iyengar, "Built-in Pattern Generation for High-Performance Circuits using Twisted-Ring Counters," in *Proc. IEEE VLSI Test Symposium*, 1999, pp. 22–27.
5. K. Chakrabarty, B.T. Murray, and V. Iyengar, "Deterministic Built-in Pattern Generation for High-Performance Circuits using Twisted-Ring Counters," *IEEE Transactions on VLSI Systems*, Vol. 8, pp. 633–636, October 2000.
6. K.-T. Cheng, S. Devadas, and K. Keutzer, "A Partial Enhanced-Scan Approach to Robust Delay-Fault Test Generation for Sequential Circuits," in *Proc. International Test Conference*, 1991, pp. 403–410.
7. C. Fagot, P. Girard, and C. Landrault, "On using Machine Learning for Logic BIST," in *Proc. International Test Conference*, 1997, pp. 338–346.

8. I. Hamzaoglu and J.H. Patel, "Test Set Compaction Algorithms for Combinational Circuits," in *Proc. International Conference on Computer-Aided Design*, 1998, pp. 283–289.
9. S. Hellebrand, H.G. Liang, and H.J. Wunderlich, "A Mixed-Mode BIST Scheme Based on Reseeding of Folding Counters," in *Proc. International Test Conference*, 2000, pp. 778–784.
10. S. Hellebrand, B. Reeb, S. Tarnick, and H. Wunderlich, "Pattern Generation for a Deterministic Scheme," in *Proc. International Conference on Computer-Aided Design*, 1995, pp. 88–94.
11. International Technology Roadmap for Semiconductors, ITRS, <http://notes.sematech.org.ntrs/PublNTRS.nsf>, 1999.
12. V. Iyengar and K. Chakrabarty, "Precedence-Based, Preemptive and Power-Constrained Test Scheduling for System-on-a-Chip," in *Proc. IEEE VLSI Test Symposium*, 2001, pp. 368–374.
13. G. Kiefer and H. Wunderlich, "Using BIST Control for Pattern Generation," in *Proc. International Test Conference*, 1997, pp. 347–355.
14. A. Krstic, K.-T. Cheng, and S.T. Chakradhar, "Testing High Speed VLSI Devices using Slower Testers," in *Proc. IEEE VLSI Test Symposium*, 1999, pp. 16–21.
15. H.K. Lee and D.S. Ha, "On the Generation of Test Patterns for Combinational Circuits," Tech. Report No. 12.93, Department of Electrical Engineering, Virginia Tech., 1993.
16. S.M. Reddy, I. Pomeranz, and S. Kajihara, "On the Effects of Test Compaction on Defect Coverage," in *Proc. IEEE VLSI Test Symposium*, 1996, pp. 430–435.
17. N.A. Toubia and E.J. McCluskey, "Synthesis of Mapped Logic for Generating Pseudorandom Patterns for BIST," in *Proc. International Test Conference*, 1995, pp. 674–682.
18. Y. Zorian, E.J. Marinissen, and S. Dey, "Testing Embedded-Core Based System Chips," in *Proc. International Test Conference*, 1998, pp. 130–143.

Shivakumar Swaminathan received the B.E. (Hons) degree from Birla Institute of Technology and Science, Pilani, India and the M.S. degree from Duke University, in 1998 and 2000, respectively. He is currently a design engineer in the PowerPC based microprocessor core design group at IBM Corporation, Research Triangle Park, NC, USA.

His research interests include Built-In Self-Test (BIST), high-performance and low-power processor design, formal verification, and novel cache architectures.

Krishnendu Chakrabarty received the B. Tech. degree from the Indian Institute of Technology, Kharagpur, in 1990, and the M.S.E. and Ph.D. degrees from the University of Michigan, Ann Arbor, in 1992 and 1995, respectively, all in Computer Science and Engineering. He is now an Assistant Professor of Electrical and Computer Engineering at Duke University.

Dr Chakrabarty is a recipient of the National Science Foundation Early Faculty (CAREER) award, the Office of Naval Research Young Investigator award and the Mercator Professor award from Deutsche Forschungsgemeinschaft, Germany. His current research projects (supported by NSF, DARPA, ONR, and industrial sponsors) are in system-on-a-chip test, embedded real-time operating systems, distributed sensor networks, and architectural optimization of microelectrofluidic systems. He has published over 80 papers in archival journals and refereed conference proceedings, and he holds a US patent in built-in self-test. He is a senior member of IEEE, a member of ACM and ACM SIGDA, and a member of Sigma Xi. He serves as Vice Chair of Technical Activities in IEEE's Test Technology Technical Council, and is a member of the program committees of several IEEE/ACM conferences and workshops.