VLSI DESIGN
2001, Vol. 12, No. 4, pp. 475–486
Reprints available directly from the publisher
Photocopying permitted by license only

Efficient Test Application for Core-Based Systems Using Twisted-Ring Counters*

ANSHUMAN CHANDRA[a,†], KRISHNENDU CHAKRABARTY[a] and MARK C. HANSEN[b]

[a]*Department of Electrical and Computer Engineering, Duke University, 130 Hudson Hall, Box 90291, Durham, NC 27708;* [b]*Delphi Delco Electronics Systems, IC Design Center, 2705 Goyer Road, P.O. Box 9005, Mail Station D18, Kokomo, IN 46904-9005*

(Received 15 August 1999; In final form 11 September 2000)

We present novel test set encoding and pattern decompression methods for core-based systems. These are based on the use of twisted-ring counters and offer a number of important advantages – significant test compression (over 10X in many cases), less tester memory and reduced testing time, the ability to use a slow tester without compromising test quality or testing time, and no performance degradation for the core under test. Surprisingly, the encoded test sets obtained from partially-specified test sets (test cubes) are often smaller than the compacted test sets generated by automatic test pattern generation programs. Moreover, a large number of patterns are applied test-per-clock to cores, thereby increasing the likelihood of detecting non-modeled faults. Experimental results for the ISCAS benchmark circuits demonstrate that the proposed test architecture offers an attractive solution to the problem of achieving high test quality and low testing time with relatively slower, less expensive testers.

Keywords: Embedded core testing; Slower-speed testers; Test set encoding; Test rate; Test vector decompression; Test-per-clock

1. INTRODUCTION

System-on-a-chip designs containing embedded cores pose a number of difficult test challenges [1]. One of these involves the reduction of test application time. Testing time is especially important for a core-based system since long test times can significantly increase cost and require design changes, thereby affecting time-to-market. Unfortunately, the testing time for a core-based system can be very high due to the fact that the test patterns for the embedded cores are applied serially *via* scan chains. In such *test-per-scan* methods, one test pattern is applied to an n-input core under test every n cycles. For embedded cores that use the same internal scan chain for applying

* This research was supported in part by the National Science Foundation under grant no. 9875324, by a contract from Delphi Delco Electronics Systems, and by an equipment grant from Sun Microsystems.
† Corresponding author. Tel.: (919) 660-5230, Fax: (919) 660-5293, e-mail: achandra@ee.duke.edu

test patterns and capturing test responses, such serialization of the test patterns is unavoidable. However, a test-per-scan method increases the testing time unnecessarily for a core that employs a boundary scan chain for test application, and uses a separate circuit for capturing test responses. While parallel access to the core in such cases is possible by multiplexing the core I/Os to chip I/Os [2], the routing and area overhead associated with such a test access mechanism can be prohibitive.

The testing time for a core-based system can be reduced by employing built-in self-test (BIST). However, BIST for logic cores is feasible only if the core vendor provides "BIST-able cores". Another problem in using BIST for core testing lies in the fact that practical (low-cost) pattern generation methods that provide high fault coverage require structural information about the IP core, either for test-point placement [3], or for carrying out fault simulation and ATPG [4]. The core vendor usually provides only a precomputed test set for the core.

In order to reduce the testing time for "non-BIST-able cores", a recently proposed test vector compression/decompression method that uses an on-chip decoder and a cyclical scan register to apply a precomputed test set T to the core [5]. Test vectors provided by the core vendor are stored in compressed form in the tester memory, and transferred serially to the core and decompressed during test application. However, a problem with this method is that it is based on a test-per-scan architecture and is therefore inefficient for cores containing boundary scan registers that do not capture test responses.

In this paper, we introduce a new test compression/decompression method based on the use of twisted-ring (Johnson) counters (TRCs) [6]. Unlike the BIST method presented in [6], the test application method described here does not require reseeding of the twisted-ring counter. For cores containing boundary scan, a *test-per-clock* architecture allows the application of a test pattern to the core every clock cycle. It imposes no performance penalty beyond what is introduced by scan design. Since a larger number of patterns are applied to the CUT every cycle, the proposed test set-up can also be used with a slower-speed tester without affecting test quality. It is becoming increasingly difficult for testers to keep up with the high frequencies needed to sufficiently test for performance-related defects in today's IC's. High-speed testers also pose greater interfacing problems during test. Moreover, the SIA National Technology Roadmap predicts that the cost of high-speed testers will exceed $20 million by 2010 [7]. Hence, test methods that can be used with slower-speed, less expensive testers are becoming especially important [8].

In contrast to a conventional test-per-scan method, the proposed method applies patterns to the core at every clock cycle, hence the test responses must be observed every clock cycle. We also show that the proposed compression method is flexible in that the TRC-based test can be designed using both fully-specified and partially-specified test sets. Finally, we compare the size of the encoded test sets (number of bits to be stored) for partially-specified patterns to the test sets obtained from ATPG programs that attempt to generate compact test sets [9]. The motivation for generating small test sets is to reduce testing time. However, the high degree of compression obtained for partially-specified test sets demonstrates that test set compaction is not always necessary – the testing time is reduced more effectively by encoding partially-specified test sets. The proposed encoding method provides significant test set compression (over 10X) for many full-scan and non-scan circuits.

The paper is organized as follows. In Section 2, we review twisted-ring counters and present our test set encoding method. As mentioned above, our encoding method is tailored towards the use of a TRC for test application. We describe our pattern application technique using TRCs and illustrate the method for a boundary scan architecture in which test responses are captured in a separate scan chain. In Section 3, we present experimental results for the ISCAS 85 [11] and

ISCAS 89 [12] benchmark circuits. Section 4 presents the conclusions and outlines directions for further research.

2. TEST SET ENCODING AND PATTERN DECOMPRESSION

In this section, we first review a recently-proposed BIST approach based on twisted-ring counters (TRCs). We then present a TRC-based test vector compression/decompression approach for embedded core testing. We describe the proposed test-per-clock pattern application technique using TRCs and illustrate the method for a boundary scan architecture in which test responses are captured in a separate scan chain.

An n-bit ring counter is a group of n flip-flops $F_1, F_2, \ldots, F_n$ 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. An n-bit TRC behaves as a counter with up to $2n$ distinct states depending on its seed. TRCs have recently been proposed for the design of BIST pattern generation circuits [6].

Test application in [6] is carried out by reconfiguring the input scan register of the circuit under test as a ring counter and a TRC. A small number of seed patterns are stored in a ROM, and for each seed, the pattern generator is clocked for $3n$ clock cycles. For the first n cycles, the pattern generator operates as a ring counter (*shift mode*) while for the remaining $2n$ cycles, it operates as a TRC (*twist mode*). A single 4-to-1 multiplexer is used to control the two-mode operation of the pattern generator. It was shown that a small number of seeds are generally sufficient to embed an arbitrary precomputed test set. The set of seeds may be viewed as an encoded test set, which is used to generate a superset of the precomputed test set during test application. A key advantage of this approach is that test-per-clock BIST pattern application is achieved without requiring any mapping logic between the scan register flip-flops and the circuit under test.

We now present another application of TRCs-test vector compression/decompression for embedded core testing that provides all the advantages of the method described in [6]. In addition, the proposed technique allows more efficient use of the TRC by removing a restriction that is inherent in the BIST architecture of [6], namely, the pattern generator can change modes (from *shift* to *twist*) only once for every seed. This restriction limits the number of patterns that can be generated from any starting state (seed) to $3n$.

We show in this paper that the entire precomputed test set can be generated from only one starting state if the pattern generator is allowed to switch modes freely, *i.e.*, at any clock cycle. In fact, any test pattern can be generated from an arbitrarily chosen initial state in at most n cycles. This observation forms the basis for the test architecture proposed in the paper. Figure 1 shows the main idea behind the pattern generator. The test set is encoded as a single-bit stream; during test application, each bit of the encoded test set determines the operation (shift or twist) to be performed during the corresponding clock cycle. The encoded test set is stored in tester memory and transferred to the IC serially during test application.

For a precomputed test set T_D with m patterns, the size of the encoded test set T_E is at most mn bits, which equals the storage requirement when no encoding is employed. However, we show later that the size of T_E is generally much smaller than mn bits. Another advantage of using the set-up shown in Figure 1 is that the patterns are applied in a test-per-clock[1] fashion without adding delays on the functional logic paths. When no encoding is employed, *i.e.*, T_D is stored in the tester as in traditional testing, a total of m patterns are applied

[1] The proposed method is test-per-clock with respect to the scan clock, as opposed to the at-speed functional clock.

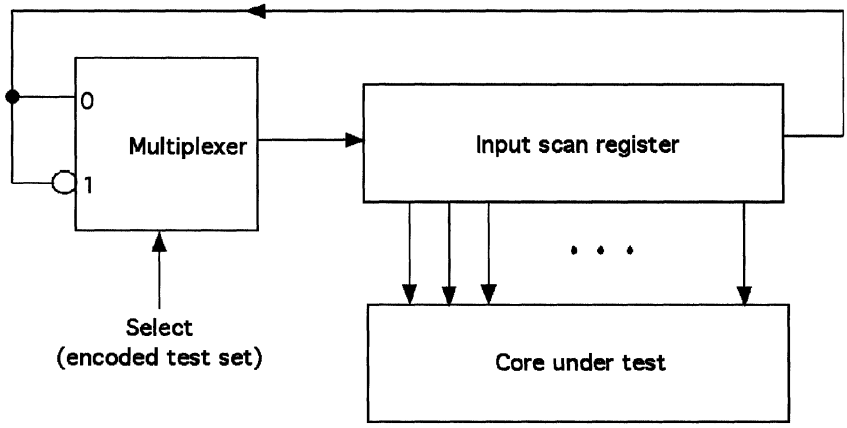

FIGURE 1 Generic pattern application using a twisted-ring counter.

to the core under test in mn cycles (test-per-scan). However, with encoding, a total of $|T_E|$ patterns are applied to the core under test in exactly $|T_E|$ cycles. Therefore, even if no compression is achieved *via* encoding in the worst case, a total of mn patterns are applied in mn cycles.

For example, consider the example T_D in Figure 2. We assume that TRC starts in the all-0 state. This can easily be achieved using a global reset. The encoded test set T_E contains only 26 bits, a saving of 47%, both in tester memory and test application time. Furthermore, using this encoded test set, a total of 26 patterns are applied to the core under test in 26 cycles, instead of the only 7 patterns that are applied in 49 cycles if T_D is stored in tester memory. The patterns from T_D that are applied to the core under test are highlighted in the figure. In general, it is not necessary to assume an all-0 initial state. Any initial state can be scanned into the TRC using the serial scan-in mechanism shown in the test architecture of Figure 3. In this paper, however, we assume that the TRC always starts in the all-0 state.

It is straightforward to show that any test pattern t can be generated from any given state S

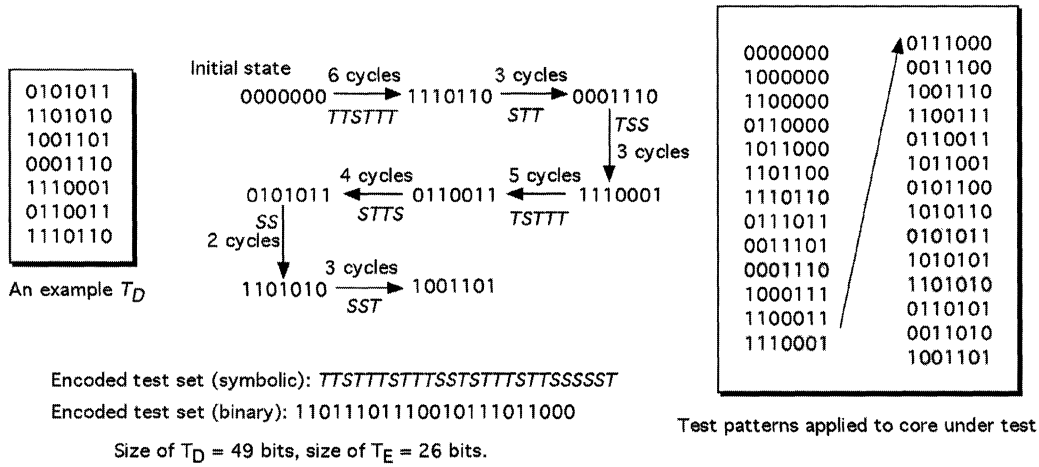

FIGURE 2 An example illustrating test set encoding and pattern application (T and S refer to twist and shift, respectively).

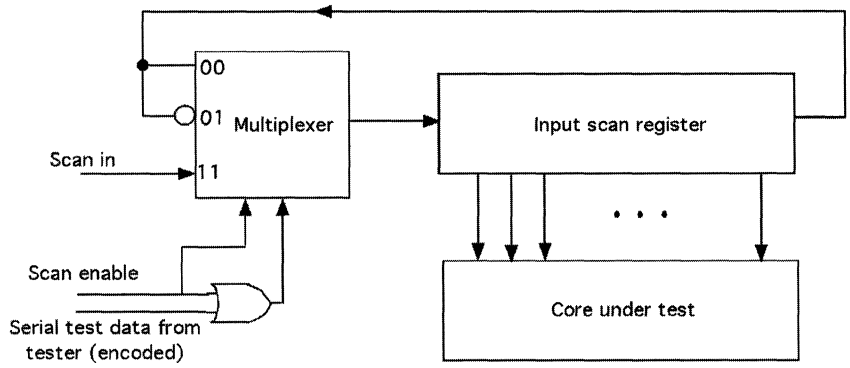

FIGURE 3 Proposed pattern application scheme using a TRC.

of an n-bit pattern generator (Fig. 1) in at most n cycles. Let $t = t_1 t_2 \cdots t_n$ and $s = s_1 s_2 \cdots s_n$. Then we can generate t from s in a minimum number of cycles by determining the smallest integer r such that $s_1 = t_{r+1}$, $s_2 = t_{r+2}, \ldots, s_{n-r} = t_n$. The test pattern can then be generated from s in exactly r cycles. Intuitively, this also implies that the TRC can always be made to generate any desired test pattern by "flushing out" s and carefully reloading the bits corresponding to t.

A block diagram of the interface between the ATE and the SOC under test is shown in Figure 4. The tester supplies patterns with frequency f_{ext}, f_{sys} is the functional clock (system) frequency, and f_{scan} is the scan clock frequency. Since a pattern is applied to the core every cycle, $f_{\text{scan}} = f_{\text{ext}}$. The on-chip clock generator is used to generate a synchronization clock to control the flow of data between ATE and the SOC, and to synchronize f_{ext} with f_{scan}. We introduce the term *test rate* for external testing to measure the number of patterns that are applied to the core under test per second. The test rate is simply f_{scan} for the TRC-based architecture. We also use the term *relative test rate*

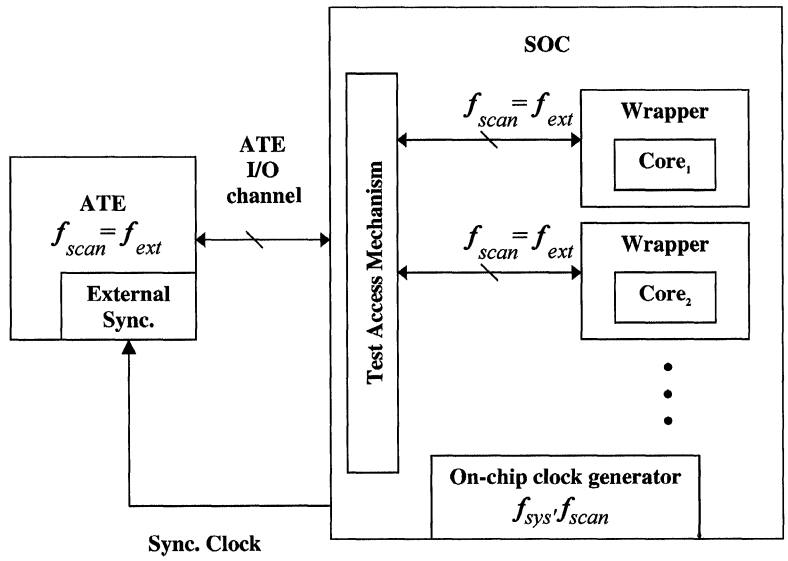

FIGURE 4 A conceptual architecture for testing a system-on-a-chip.

to measure the number of patterns applied to the core under test per tester cycle. For the pattern application scheme in Figure 3, the relative test rate is the maximum value possible, *i.e.*, 1. Note that for a test-per-scan scheme, the test rate is only f_{scan}/n and the relative test rate is only $1/n$. Let αmn be the size of the encoded test set T_E. In other words, $\alpha = |T_E|/|T_D|$ is directly proportional to the amount of compression achieved using the proposed encoding method.

Test-per-clock pattern application is therefore achieved using a TRC without adding any additional logic on the critical paths between the flip-flops and the inputs of the core under test. Hence, as in test-per-scan methods [14, 15] , the proposed method imposes no additional performance degradation beyond scan. This is in contrast to most test-per-clock methods, which require mapping logic between the flip-flops and the circuit under test. Furthermore, the test-per-clock approach allows us to use slower, low-cost testers. For example, in order to apply all the test patterns in time T, a test-per-scan method requires the tester to run at frequency such that $f_{scan} = mn/T$. With our test-per-clock approach, the testing can be carried out in the same amount of time using a tester that runs at frequency such that $f_{scan} = \alpha mn/T$. Moreover, an additional $m(\alpha n - 1)$ patterns are applied to the circuit in the same amount of time. This increases the likelihood of detecting non-modeled faults that are not explicitly targeted by T_D.

As in any test-per-clock approach, response monitoring in the proposed method must be done every tester clock cycle. This can be carried out using a combination of space compaction [16, 17] and multiple-input signature register. The response monitoring logic can be designed from the responses of the core under test to the patterns in T_D and to the additional patterns that are applied by the TRC.

We now address the issue of determining the encoded test set T_E from the given precomputed test set T_D. We first note that for full-scan cores, since the patterns can be reordered, the encoding problem can be formulated as the well-known traveling salesman problem (TSP) on a weighted directed graph. Every pattern in T_D corresponds to a node in a directed graph G. An additional node s in this graph corresponds to the all-0 initial state of the pattern generator. The weight of an edge (x, y) in G equals the minimum number of cycles required to generate $y = y_1 y_2 \cdots y_n$ from $x = x_1 x_2 \cdots x_n$. This is determined from the smallest $r \leq n$ such that $x_1 = y_{r+1}$, $x_2 = y_{r+2}, \ldots, x_{n-r} = y_n$. It can now be easily seen that the minimum-size encoding of T_D corresponds to a minimum-cost Hamiltonian path (TSP) in G that starts at s.

Since the traveling salesman problem is NP-complete [10], we use a simple greedy algorithm to carry out the encoding. Starting from the all-0 initial state S, we generate the pattern in T_D that is at the least "distance" from S. (The distance of y from x is given by the weight of the edge from x to y in G.) We then continue this process until all the patterns in T_D have been generated. Figure 2 illustrates the greedy algorithm for an example test set. The complexity of this algorithm is $O(m^2)$, and the complexity of generating the graph is $O(m^2 n)$.

The encoding algorithm can also be applied to test sets containing partially-specified patterns (test cubes). In fact, significantly more compression can be expected if T_D is partially-specified. In order to illustrate the encoding for test cubes, we introduce the notion of compatibility between the bits x_i and y_j of test cubes x and y. We define x_i and y_j to be *compatible* if either (i) both x_i and y_j are specified and equal, or (ii) at least one of these two bits is a don't care. Once again, we start from the all-0 s initial state and follow a greedy strategy of choosing a pattern at the least distance, *i.e.*, a node in G connected by an outgoing edge from s with the least weight. The weight $w(x, y)$ of the edge from x to y is determined as follows: $w(x, y) = r$, where r is the smallest integer such that x_i and y_{r+i} are compatible, $1 \leq i \leq n - r$. Note that the *don't*-cares of every test cube generated from an initial state are always uniquely mapped to 1s and 0s.

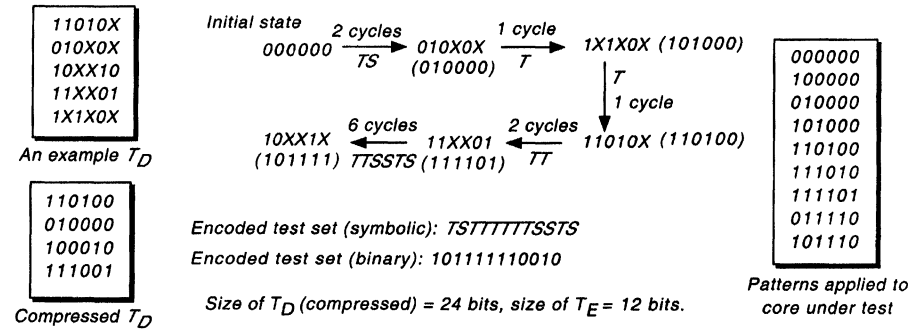

FIGURE 5 An example illustrating compression and pattern generation for a partially-specified test set.

As an example, let us refer to Figure 5. Starting from the all-0 initial states, the first pattern from T_D that is applied to the core under test is $t_1 = 010X0X$. However, since the starting state is completely-specified, the two don't-care bits of t_1 are mapped to 0s. The state of the pattern generator is now 010000 (shown in the figure), and the next pattern from T_D that is applied to the core under test is therefore $t_2 = 1X1X0X$. The three don't cares of this test cube are uniquely mapped to 0s. In general, this mapping is not unique; therefore long runs of 1s and 0s can be obtained by suitably assigning values to the don't-care bits. We have implemented the encoding algorithm in C++ and the experimental results for the ISCAS benchmark circuits are reported in Section 4.

It is possible that a few of the patterns generated by the TRC may be repetitions of patterns already applied to the core. However, this seems to occur rarely. For example, no patterns are repeated in the examples of Figure 2 and Figure 4. A general characterization of the sequences generated by the test architecture of Figure 4 remains an interesting open problem.

The example in Figure 4 illustrates another key advantage of the proposed compression/decompression approach. A test set containing test cubes is usually compressed to generate a compact, completely-specified test set that is stored in tester memory. This is typically carried out during automatic test pattern generation (ATPG) [9]. In the example of the test set of Figure 4, the number of bits required for storage can be reduced in this way from 30 to 24. This also reduces the testing time by 6 cycles for a test-per-scan testing scheme. On the other hand, if encoding is performed on the partially-specified test set, only 12 bits are required for storage. This represents a saving of 50% in tester memory and test application time. The patterns applied to the core under test are also shown in Figure 4 (the patterns from T_D are highlighted).

3. EXPERIMENTAL RESULTS

In this section, we evaluate the proposed test set compression/decompression method for the ISCAS benchmark circuits. We use the following measures in our evaluation: tester memory requirement, test application time, number of patterns applied to the core under test, and the test rate. Test data compression is not the only benefit of using the proposed pattern application scheme. As our experimental results demonstrate, we achieve significantly higher test rates for test-per-clock architectures.

Table I presents the experimental results for fully-specified test sets generated using the Mintest ATPG program [9]. These experiments were carried out using a Sun Ultra 10 workstation with a 333 MHz processor and 128 MB of DRAM. We compare the proposed test-per-clock scheme with

TABLE I Experimental results for the full-scan ISCAS circuits with completely-specified test patterns generated using Mintest [9]

			Proposed method			Test-per-scan	
Circuit	n	m	α	Testing time (cycles)	No. of patterns applied	Testing time (cycles)	No. of patterns applied
c432	36	27	0.8457	822	822	972	27
c499	41	52	0.8908	1899	1899	2132	52
c880	60	16	0.9750	936	936	960	16
c1355	41	84	0.9760	3361	3361	3444	84
c1908	33	106	0.8385	2933	2933	3498	106
c2670	233	44	0.9814	10061	10061	10252	44
c3540	50	84	0.8917	3745	3745	4200	84
c5315	178	37	0.9700	6388	6388	6586	37
c6288	32	12	0.8099	311	311	384	12
c7552	207	73	0.9766	14757	14757	15111	73
s208	19	27	0.7837	402	402	513	27
s298	17	23	0.7648	299	299	391	23
s344	24	13	0.8334	260	260	312	13
s349	24	23	0.8494	265	265	312	23
s382	24	25	0.7934	476	476	600	25
s386	13	63	0.5825	477	477	819	63
s400	24	24	0.8125	468	468	576	24
s420	35	43	0.8971	1350	1350	1505	43
s444	24	24	0.8334	480	480	576	24
s510	25	54	0.7860	1061	1061	1350	54
s526	24	49	0.7241	838	838	1176	49
s641	54	21	0.9224	1046	1046	1134	21
s713	54	21	0.9400	1066	1066	1134	21
s820	23	93	0.7420	1587	1587	2139	93
s832	23	94	0.7392	1598	1598	2162	94
s838	67	75	0.9521	4784	4784	5025	75
s953	45	76	0.8886	3039	3039	3420	76
s1196	32	113	0.8067	2917	2917	3616	113
s1238	32	121	0.7950	3078	3078	3872	121
s1423	91	20	0.9566	1741	1741	1820	20
s1488	14	101	0.5821	823	823	1414	101
s1494	14	100	0.5486	768	768	1400	100
s5378	214	97	0.9704	20413	20413	20758	97
s9234	247	105	0.9747	25278	25278	25935	105
s13207	700	234	0.9899	162152	162152	163800	234
s15850	611	95	0.9901	57472	57472	58045	95

a conventional test-per-scan scheme for these test sets. The fractional reduction in tester memory and testing time is given by $1-\alpha$. Further compression can be achieved by applying run-length coding (or other coding techniques) to the encoded test set, with the corresponding overhead of on-chip decoders. While the encoding procedure reduces the tester memory (and thereby testing time) in all cases, the comparison becomes more striking when we examine the number of test patterns applied to the core under test during this period. The number of patterns applied to the core (and hence the test rate) is increased several orders of magnitude. The test quality is therefore significantly improved due to the increased likelihood of detecting non-modeled faults.

We next compare the proposed method with the compression/decompression method of [5]. The compression reported in [5] is higher than that obtained by us using compact test sets generated by Mintest. However, the results reported in [5] were obtained using test sets with a considerably larger number of patterns. Since we do not have access to the test sets used in [5], we carry out a comparison using test sets generated by Atalanta [13]. The sizes of the test sets generated by Atalanta are similar to those used in [5], and the compression achieved using by our method improves significantly for the larger test sets, thereby allowing a reasonable comparison.

Further compression can be expected by applying run-length coding to T_E. The results in Table II show that if the proposed encoding method is used with Atalanta test sets, the tester memory is reduced over [5] in a number of cases. More importantly, the test-per-clock architecture allows a much larger of patterns to be applied to the core under test.

We next evaluate the proposed test-per-clock method for partially-specified patterns (test cubes). We first use test patterns obtained by instrumenting the Atalanta program, as well as test cubes from the Mintest program developed at University of Illinois. Table III shows that considerable test compression (over 10X in many cases) is obtained for all these test sets. The number of bits stored, *i.e.*, the size of the encoded test, is equal to the testing time (in cycles). It is also equal to the number of patterns applied to the core under test.

Surprisingly, these results also demonstrate that test set compaction may not always be necessary during automatic test pattern generation. We compare the amount of test data obtained using ATPG compaction to the amount of test data obtained by encoding the partially-specified test sets. (The amount of ATPG-compacted test data for Mintest and Atalanta are obtained from Tables II and III, respectively.) The circuits for which encoding of test cubes leads to less test data are shaded in Table III. We did not generate the compacted test sets using Atalanta for the full-scan versions of the ISCAS 89 circuits hence a comparison was not possible in this case.

Finally, we note that $\alpha \geq 1/n$, since the size of the encoded test set is at least m bits. Quite remarkably, this lower bound is nearly achieved for several circuits in Table III. For example, for s344 and s349, the lower bound on α is 0.0417. Using the Mintest test sets, we obtain $\alpha = 0.0437$ and 0.0494, respectively.

TABLE II Comparison of TRC-based pattern generation for completely-specified Atalanta test sets with test-per-scan method of [5]

	Proposed method					Test per scan [5]			
Circuit	m	α	$\lvert T_E \rvert$	Testing time (cycles)	No. of patterns applied	m	$\lvert T_E \rvert$	Testing time (cycles)	No. of patterns applied
c432	49	0.84	1489	1489	1489	59	1608	1608	59
c499	53	0.87	1898	1898	1898	58	1449	1449	56
c880	53	0.93	2956	2956	2956	80	3930	3930	80
c1355	84	0.85	2930	2930	2930	103	2910	2910	103
c1908	144	0.79	3791	3791	3791	130	3159	3159	130
c2670	103	0.97	23279	23279	23279	133	24405	24405	133
c3540	179	0.86	7678	7678	7678	198	7752	7752	198
c5315	122	0.95	20775	20775	20775	214	30501	30501	214
c6288	40	0.86	1100	1100	1100	36	912	912	36
c7552	215	0.96	42952	42952	42952	271	42528	42528	271

TABLE III Experimental results (test-per-clock) for partially-specified test sets

	Atalanta test sets				Mintest test sets			
Circuit	m	α	No. of bits stored for proposed encoding method	Amount of test data obtained using ATPG compaction	m	α	No. of bits stored for proposed encoding method	Amount of test data obtained using ATPG compaction
c432	128	0.3273	1508	1764	211	0.2387	1813	972
c499	93	0.3913	1492	2173	114	0.6569	3070	2132
c880	355	0.2605	5548	3180	406	0.1676	4082	640
c1355	143	0.8136	4770	3444	198	0.6327	5136	3444
c1908	434	0.6919	9909	4752	262	0.5749	4970	3498
c2670	639	0.2652	39483	23999	775	0.1874	33839	10252
c3540	–	–	–	–	891	0.1306	5818	4203
c6288	201	0.5176	3329	1280	202	0.5680	3671	384
c5315	1213	0.3094	66789	21716	1552	0.2018	55748	6586
c7552	845	0.5030	87929	44505	1133	0.2929	68694	15111
s208	64	0.2459	299	–	81	0.1391	214	513
s298	93	0.0981	155	–	127	0.0649	140	391
s344	85	0.1746	356	–	129	0.0437	135	312
s349	85	0.1697	346	–	126	0.0494	149	552
s382	102	0.1336	327	–	118	0.1035	293	600
s386	102	0.3824	507	–	120	0.2231	348	819
s400	96	0.1225	289	–	120	0.0796	229	576
s420	129	0.2012	908	–	161	0.1321	744	1505
s444	107	0.1036	266	–	128	0.0889	273	576
s510	103	0.2804	722	–	130	0.0687	223	1350
s526	160	0.1857	713	–	207	0.1045	519	1176
s641	200	0.0987	1065	–	231	0.1395	1740	1134
s713	192	0.1396	1447	–	220	0.1398	1661	1134
s820	195	0.2973	1333	–	253	0.1219	709	2139
s832	199	0.2740	1254	–	255	0.1252	734	2162
s838	257	0.1456	2506	–	319	0.1265	2703	5025
s953	227	0.1513	1545	–	255	0.1309	1502	3420
s1196	291	0.1773	1651	–	346	0.3196	3539	3616
s1238	164	0.1916	1005	–	360	0.3431	3953	3872
s1423	89	0.1334	1080	–	524	0.1285	6127	1820
s1488	252	0.4159	1467	–	317	0.1370	608	1414
s1494	250	0.2666	933	–	311	0.1282	558	1400
s5378	–	–	–	–	1459	0.0827	25821	20758
s9234	–	–	–	–	1929	0.1584	75471	25935
s13207	–	–	–	–	3237	0.0634	143658	163800
s15850	–	–	–	–	3290	0.0779	156593	58045

4. CONCLUSIONS

We have presented a novel test vector compression method and decompression architecture for embedded core testing. These are based on the use of a twisted-ring counter, which can operate in both the shift and twist modes. The serial bit stream that controls the operation of the counter comprises the encoded test set. The proposed method offers a number of important advantages – significant test compression (over 10X in many cases), less tester memory and reduced testing time, the ability to use a slow tester without compromising test quality or testing time, and

no performance degradation for the core under test.

The decompressed patterns are applied test-per-clock, thereby applying a large number of patterns to the core under test and increasing the likelihood of detecting non-modeled faults. Experimental results for the ISCAS benchmark circuits demonstrate that the proposed test architecture offers an attractive solution to the problem of achieving high test quality and low testing time with relatively slower, low-cost testers. The experimental results also show that the encoded test sets obtained from partially-specified test sets (test cubes) are often smaller than the compacted test sets generated by automatic test pattern generation programs. The proposed scheme has a limitation that it can be only applied to register based designs such as the designs which employ built-in logic-block observation (BILBO) registers.

References

[1] Zorian, Y., Marinissen, E. J. and Dey, S. (1998). "Testing embedded-core based system chips", *Proc. International Test Conference*, pp. 130–143.

[2] Immaneni, V. and Raman, S. (1990). "Direct access test scheme-design of block and core cells for embedded ASICs", *Proc. 1990 International Test Conference*, pp. 488–492.

[3] Cheng, K.-T. and Lin, C.-J. (1995). "Timing-driven test point insertion for full-scan and partial-scan BIST", *Proc. 1995 International Test Conference*, pp. 506–514.

[4] Fagot, C., Girard, P. and Landrault, C. (1997). "On using machine learning for logic BIST", *Proc. International Test Conference*, pp. 338–346.

[5] Jas, A. and Touba, N. A. (1998). "Test vector decompression *via* cyclical scan chains and its application to testing core-based designs", *Proc. International Test Conference*, pp. 458–464.

[6] Chakrabarty, K., Murray, B. T. and Iyengar, V. (1999). "Built-in test pattern generation for high performance circuits using twisted-ring counters", *Proc. VLSI Test Symposium*, pp. 22–27.

[7] http://notes.sematech.org/97pelec.htm, The National Technology Roadmap for Semiconductors (NTRS), Silicon Industry Association (SIA), 1997.

[8] Agrawal, V. D. and Chakraborty, T. J. (1995). "High-performance circuit testing using slow-speed testers", *Proc. International Test Conference*, pp. 302–310.

[9] Hamzaoglu, I. and Patel, J. H. (1998). "Test set compaction algorithms for combinational circuits", *Proc. International Conference on CAD*, pp. 283–289.

[10] Garey, M. S. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, W. H. and Company, New York.

[11] Brglez, F. and Fujiwara, H. (1985). A neutral netlist of 10 combinational benchmark circuits and a target simulator in Fortran, *Proc. International Symposium on Circuits and Systems*, pp. 695–698.

[12] Brglez, F., Bryan, D. and Kozminski, K. (1989). "Combinational profiles of sequential benchmark circuits", *Proc. 1989 Int. Symposium on Circuits and Systems*, pp. 1929–1934.

[13] Lee, H. K. and Ha, D. S., "On the Generation of Test Patterns for Combinational Circuits", *Tech. Rep. No. 12_93, Dept. of Electrical Eng.*, Virginia Poly. Institute and State Univ.

[14] Rajski, J., Tyszer, J. and Zacharia, N. (1998). "Test data decompression for multiple scan designs with boundary scan", *IEEE Transactions on Computers*, **47**(11), 1188–1200.

[15] Zacharia, N., Rajski, J., Tyszer, J. and Waicukauski, J. A. (1996). "Two-dimensional test data decompressor for multiple scan designs", *Proc. International Conference*, pp. 186–194.

[16] Chakrabarty, K., Murray, B. T. and Hayes, J. P. (1998). "Optimal zero-aliasing space compaction of test responses", *IEEE Transactions on Computers*, **47**(11), 1171–1187.

[17] Savir, J., "Shrinking wide compressors", *IEEE Transactions on Computer-Aided Design*, **14**, 1379–1387, Nov., 1995.

Authors' Biographies

Anshuman Chandra received the B.E. degree in Electrical engineering from the University of Roorkee, Roorkee, India, in 1998, M.S. degree in Electrical and Computer Engineering from Duke University, Durham, USA, in 2000, and is currently enrolled in the Ph.D. program at Duke University. His research interests are in the fields of VLSI Design, Digital Testing and Computer Architecture. He is a recipient of TTTC James Beausang Student Paper award for DFT for a paper published in *Proc. 2000 IEEE VLSI Test Symposium*. He is currently working in the areas of test set compression/decompression, embedded core testing and built-in self test (BIST). He is a student member of IEEE.

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 Assistant Professor of Electrical and Computer

Engineering at Duke University. Dr. Chakrabarty is a recipient of the National Science Foundation's Early Faculty (CAREER) award, the office of Naval Research Young Investigator Award and the Mercator Professor award from the Deutsche Forschungsgemeinschaft, Germany for carrying out research at the University of Potsdam during 2000–2001. His current research projects (supported by NSF, ONR, DARPA 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 65 papers in archival journal and refereed conference proceedings, and he holds a US patent in built-in self-test. He is a senior member of IEEE, member of Sigma Xi, 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.

Mark C. Hansen supervises the IC Design Automation Operations groups at Delphi Delco Electronics Systems in Kokomo, Indiana. His research interests include functional-level fault modeling, design for test, ATPG, BIST, and analog test. He holds a dozen patents. Hansen received a BSEE from Michigan State University, and MSEE from Carnegie Mellon University, and a Ph.D. in Computer Science and Engineering from University of Michigan, Ann Arbor. He is a member of the IEEE and the IEEE Computer Society.