Storage-Based Logic Built-In Self-Test with Cyclic Tests

Irith Pomeranz

***Abstract*— Logic built-in self-test ($LBIST$) eliminates the need for external test data, and thus facilitates in-field testing. Instead of pseudo-random tests that are typically produced by on-chip test generation logic, storage-based $LBIST$ uses deterministic test data entries, which are stored on-chip, for applying tests that are closer to deterministic tests. This article describes a storage-based $LBIST$ approach that uses a unique type of scan-based tests referred to as cyclic tests. Cyclic tests have cyclic scan enable and scan in sequences, with a higher proportion of functional capture cycles compared with conventional scan-based tests. This results in more clock cycles where fault effects can be captured in the flip-flops. The improved fault detection capabilities this provides helps balance the number of applied tests and the storage requirements. Experimental results for benchmark circuits demonstrate the effectiveness of cyclic tests.**

***Index Terms*— Cyclic tests, full-scan, logic built-in self-test ($LBIST$), on-chip test generation.**

I. Introduction

Logic built-in self-test ($LBIST$) eliminates the need for external test data, and thus facilitates in-field testing [1]-[16]. The elimination of external test data also enhances security [5], and allows test application to be carried out at-speed.

On-chip test generation logic under $LBIST$ is typically based on producing pseudo-random tests [1]. This is motivated by the simplicity of the hardware needed for generating pseudo-random test data. To improve the fault coverage, one of several approaches may be used. (1) Test-points may be inserted into the circuit. (2) Pseudo-random tests may be modified, e.g., by using weights, bit-flipping or bit-fixing logic, to improve the fault coverage. (3) The on-chip test generator may be initialized multiple times using multiple seeds that are stored on chip or derived from seeds that are stored on chip by bit complementation. (4) Pseudo-random tests may be supplemented by deterministic tests. The universal use of test data compression supports solutions where compressed tests are stored on-chip and decompressed by the on-chip decompression logic to produce deterministic tests and tests that are derived from them by bit or scan slice complementation.

The storage-based $LBIST$ solutions described in [4], [14] and [15] store uncompressed deterministic test data entries, e.g., scan vectors, on-chip. They combine stored test data entries on-chip to form tests that are close (but not necessarily equal) to deterministic tests. Every stored test data entry may be used multiple times as part of different tests to create a large number of substantially different tests, all consisting of deterministic test data, from which a subset can be selected. This allows the volume of stored test data to be reduced while achieving complete fault coverage. In [14] and [15], complete fault coverage is achieved for stuck-at and single-cycle gate-exhaustive faults. Combining of test data entries may be performed pseudo-randomly (using linear-feedback shift-registers) or deterministically (by storing additional data indicating how to combine test data entries to form tests). With deterministic combinations the number of tests is smaller than with random combinations, but the storage requirements are higher.

This article describes a storage-based $LBIST$ approach that uses a unique type of scan-based tests, referred to as cyclic tests, with the objective of balancing the number of tests that need to be applied and the storage requirements. A cyclic test has cyclic scan enable and scan in sequences. With a cyclic scan enable sequence, a cyclic test has more functional capture cycles than a conventional scan-based test. This results in more clock cycles where fault effects can be captured in the flip-flops, and more opportunities for faults to be detected. Specifically, if a fault f is activated during a scan shift cycle, the fault effects disappear without detecting the fault. In [17]-[19], such a fault f is detected by inserting observation points on next-state variables. With cyclic tests, instead of observation points, an increased number of functional capture cycles is used for capturing fault effects in the flip-flops, and allowing faults to be detected.

In general, the use of cyclic tests allows sequences of scan shift cycles and functional capture cycles that are not used by conventional scan-based tests. This is also the case with the approach referred to as transparent-scan [20]. An alternative to cyclic tests is to use random tests with random scan enable sequences. In [6], such an $LBIST$ approach is described for circuits with multiple scan chains, and multiple scan enable inputs. The scan enable inputs are controlled independently in [6] to allow different scan chains to operate in different modes.

In this article, a single scan enable input ensures that all the scan flip-flops operate together in scan shift or functional capture mode. Cyclic tests are formed from deterministic test data to control the number of tests. The use of cyclic tests, instead of general transparent-scan tests, limits the storage requirements by avoiding the storage of scan enable sequences. A software procedure is described that accepts a conventional scan-based test set C, and determines which parts of C will be stored on-chip and used for generation of cyclic tests.

Experimental results for single stuck-at faults in benchmark circuits demonstrate that the use of cyclic tests allows complete single stuck-at fault coverage to be achieved using smaller numbers of applied tests and storage requirements similar to [15]. In [15], pseudo-random combinations of deterministic test data are used for forming conventional scan-based tests. The smaller number of cyclic tests is important when test application under $LBIST$ occurs at system startup or during intervals where the system is idle. These scenarios occur during in-field testing of systems with high reliability requirements.

The use of cyclic tests has a computational cost similar to transparent-scan [20] since it requires fault simulation to consider all the clock cycles of a test. Considering a circuit with K flip-flops in its longest scan chain, a single-cycle scan-based test has $2K + 1$ clock cycles: K scan shift cycles for a scan-in operation, a functional capture cycle, and K scan shift cycles for a scan-out operation. Of these clock cycles, only the functional capture cycle needs to be simulated. A cyclic test with $2K + 1$ clock cycles does not rely on K scan shift cycles to bring the circuit to a known state or observe a known state. This allows it to use a higher proportion of functional capture cycles and detect more faults with every test. The computational cost is that all the clock cycles of the test (both scan shift and functional capture cycles) need to be simulated. In addition, fault simulation is carried out for the circuit with the scan logic to ensure correct fault simulation of both types of clock cycles. The $LBIST$ approach with cyclic tests is applied to logic blocks for which the fault simulation effort of cyclic tests is manageable using available academic fault simulation tools. Commercial tools are able to simulate arbitrary sequences of functional capture and scan shift cycles [21]. This capability can potentially be extended to consider transparent-scan sequences and cyclic tests.

By simulating all the clock cycles of a test, including both scan shift and functional capture cycles, and using the circuit with the scan logic, fault simulation yields well-defined output responses that can be used for fault detection as well as diagnosis.

Irith Pomeranz is with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, U.S.A. (e-mail: pomeranz@ecn.purdue.edu).

The work was supported in part by NSF Grant No. CCF-2041649.

TABLE I
CYCLIC TEST CORRESPONDING TO c_i

	(a) $m = 6, l = 5$			(b) $m = 3, l = 2$		
u	A	a_{en}	a_{in}	A	a_{en}	a_{in}
0	01	1	0	01	1	0
1	01	1	1	01	1	1
2	01	1	1	01	0	0
3	01	1	0	01	1	1
4	01	1	1	01	1	0
5	01	0	0	01	0	1
6	01	1	1	01	1	0
7	01	1	1	01	1	1
8	01	1	0	01	0	0
9	01	1	1	01	1	1
10	01	1	0	01	1	0

The article is organized as follows. The on-chip test generation process is described in Section II. A software procedure for determining the test data stored on-chip is described in Section III. Experimental results for benchmark circuits are presented in Section IV.

II. ON-CHIP TEST GENERATION

This section describes the on-chip test generation process and the hardware required for implementing it.

A. Preliminaries

The main part of the on-chip test generation logic is a memory for storing deterministic test data, which are obtained from a scan-based test set denoted by C. For simplicity of discussion, C is assumed to contain single-cycle tests. A single-cycle test $c_i = \langle s_i, v_i \rangle \in C$ consists of a scan-in state s_i, and a primary input vector v_i.

For a circuit with K flip-flops in its longest scan chain, application of c_i to the circuit starts with K scan shift cycles where s_i is scanned in. A functional capture cycle follows the scan-in operation. During this clock cycle, v_i is applied to the primary inputs. The test ends with K scan shift cycles for a scan-out operation. Thus, application of c_i to the circuit takes $2K + 1$ clock cycles.

The cyclic test t_i corresponding to c_i, which is the cyclic test that duplicates c_i, also has $2K + 1$ clock cycles. Denoting the scan enable input by a_{en}, a scan shift cycle has $a_{en} = 1$, and a functional capture cycle has $a_{en} = 0$. The sequence applied to a_{en} under t_i is 1...101...1, with two subsequences of K consecutive 1's.

Let the set of scan chain inputs be a_{in}. During the first K clock cycles of t_i, s_i appears on a_{in}, one scan vector per clock cycle. Additional scan vectors are not specified by c_i. To obtain a cyclic test with $2K + 1$ clock cycles, the sequence of scan vectors on a_{in} is repeated. In addition, the primary input vector v_i is applied to the primary inputs during all the clock cycles of t_i.

For illustration, let $c_i = \langle 01101, 01 \rangle$ be a scan-based test for a circuit with two primary inputs and $K = 5$ flip-flops in a single scan chain. The cyclic test t_i corresponding to c_i has $2K + 1 = 11$ clock cycles. For every clock cycle $0 \leq u < 11$, Table I(a) shows the primary input vector, the value of the scan enable input, and the value of the scan chain input under columns A, a_{en}, and a_{in}, respectively.

All the cyclic tests considered in this article consist of $2K + 1$ clock cycles. This is the minimum number of clock cycles required to guarantee that every scan-based test c_i can be duplicated on-chip. In addition, all the choices made with respect to the on-chip generation of cyclic tests ensure that the on-chip logic is simple, and can duplicate c_i if necessary. These choices are described next.

B. Scan Enable Sequences

All the scan enable sequences are obtained using the same logic consisting of a down-counter. By initializing the down-counter to different values, the scan enable sequences differ in the number of functional capture cycles they contain. This is important since additional functional capture cycles allow additional faults to be detected using cyclic tests.

TABLE II
SCAN ENABLE SEQUENCES

m	$(1)^{m-1}0$	SE_m										
1	0	0	0	0	0	0	0	0	0	0	0	0
2	10	1	0	1	0	1	0	1	0	1	0	1
3	110	1	1	0	1	1	0	1	1	0	1	1
4	1110	1	1	1	0	1	1	1	0	1	1	1
5	11110	1	1	1	1	0	1	1	1	1	0	1
6	111110	1	1	1	1	1	0	1	1	1	1	1

TABLE III
SCAN IN SEQUENCES

l	$p_{i,l}$	$SI_{i,l}$										
1	0	0	0	0	0	0	0	0	0	0	0	0
2	01	0	1	0	1	0	1	0	1	0	1	0
3	011	0	1	1	0	1	1	0	1	1	0	1
4	0110	0	1	1	0	0	1	1	0	0	1	1
5	01101	0	1	1	0	1	0	1	1	0	1	0

To describe the scan enable sequences produced by the on-chip test generation logic, the sequence $aa...a$, with M repetitions of a, is denoted by $(a)^M$. A sequence of length $2K + 1$ with a cycle of length m is specified by providing only one copy of its cycle. The cycle is repeated to form a sequence of length $2K + 1$.

Using this notation, the scan enable sequence of a scan-base test c_i has cycle $(1)^K0$. Repeating the cycle to obtain a sequence of length $2K + 1$ yields the sequence $(1)^K0(1)^K$. A down-counter needs to be initialized to K, and produce the value 0 when the count reaches zero. Initialization is repeated twice.

Up to $K + 1$ different scan enable sequences are obtained by initializing the counter with a value $m - 1$, where $1 \leq m \leq K + 1$. The scan enable sequence for m has cycle $(1)^{m-1}0$. This scan enable sequence is denoted by SE_m. It requires the counter to be initialized to $m-1$ repeatedly $\lceil (2K+1)/m \rceil$ times after the count reaches zero. The number of functional capture cycles for SE_m is $\lfloor (2K+1)/m \rfloor$.

For illustration, the scan enable sequences of length $2K + 1 = 11$ obtained with $K = 5$ and $1 \leq m \leq K + 1$ are shown in Table II.

C. Scan In and Primary Input Sequences

Similar to the scan enable sequences, the scan in sequences produced by the on-chip test generation logic are also cyclic. The cycle is denoted by $1 \leq l \leq K$. The sequences are obtained from a scan-based test $c_i = \langle s_i, v_i \rangle \in C$. A scan in sequence $SI_{i,l}$, for $1 \leq l \leq K$, is obtained from c_i as follows.

Let the prefix $p_{i,l}$ of s_i consist of the first l scan vectors of s_i. The scan in sequence $SI_{i,l} = p_{i,l}$ is obtained by repeating $p_{i,l}$ to obtain a sequence of length $2K + 1$.

For illustration, suppose that the circuit has a single scan chain of length $K = 5$, and $s_i = 01101$. Table III shows the cyclic scan in sequences obtained with $1 \leq l \leq K$.

For a parameter denoted by l_i, the on-chip memory will contain a prefix p_i of length l_i of s_i. A down-counter initialized to $l - 1$ for $l \leq l_i$, and initialized again after it reaches zero, produces the scan in sequence $SI_{i,l} = p_{i,l}$ of length $2K + 1$.

In addition, the primary input vector v_i of c_i is stored on-chip. When a cyclic test based on c_i is generated on-chip, the primary input vector v_i is held constant for the duration of the test.

D. Cyclic Tests

Based on the discussion thus far, a scan-based test $c_i = \langle s_i, v_i \rangle \in C$ is associated with several parameters to determine which cyclic

TABLE IV
PARAMETERS FOR CYCLIC TESTS

parameter	meaning
$1 \leq l_i \leq K$	length of cycle of scan in sequence
$1 \leq m_i \leq K+1$	length of cycle of scan enable sequence
v_i	primary input vector
p_i	prefix of length l_i of scan-in state s_i

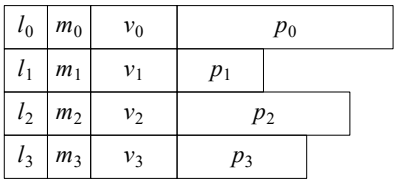

Fig. 1. On-chip memory.

tests will be applied to the circuit The parameters are shown in Table IV. They are stored on-chip as a single entry $e_i = \langle l_i, m_i, v_i, p_i \rangle$. To ensure that every entry is as effective as possible in detecting faults, and as few entries as possible are needed, an entry e_i is used for applying $l_i \cdot m_i$ tests, with $1 \leq l \leq l_i$ and $1 \leq m \leq m_i$. The test $t_{i,l,m}$ uses SE_m, $SI_{i,l}$ and v_i.

For illustration, let $c_i = \langle 01101, 01 \rangle$ be associated with $m_i = 3$ and $l_i = 2$. Cyclic tests are formed by using every SE_m with $1 \leq m \leq 3$ from Table II, every $SI_{i,l}$ with $1 \leq l \leq 2$ from Table III, and $v_i = 01$. The test with $m = 3$ and $l = 2$ is shown in Table I(b).

The test from Table I(a) is obtained from c_i using $m = m_i = 6$ and $l = l_i = 5$. In general, the option of using $m_i = K + 1$ and $l_i = K$ ensures that a cyclic test t_i corresponding to a scan-based test c_i can be applied by the on-chip test generation logic. This ensures that complete fault coverage can be achieved.

E. On-Chip Storage of Test Data and Applied Tests

The on-chip memory required for storage of test data is illustrated by Figure 1. Four scan-based tests from C contribute test data that are used for generating cyclic tests in Figure 1.

In general, a subset of scan-based tests from C contributes test data for on-chip test generation. The subset is denoted by C_{tg}. The size of the memory is determined as follows.

Let $l_{max} = max\{l_i : c_i \in C_{tg}\}$, and $m_{max} = max\{m_i : c_i \in C_{tg}\}$. For a circuit with N primary inputs, and H scan chains, the number of bits required for storing test data is $\sum\{\lceil log_2(l_{max}) \rceil + \lceil log_2(m_{max}) \rceil + N + Hl_i : c_i \in C_{tg}\}$.

In addition to the memory, the on-chip test generation logic requires two down-counters, and an $LFSR$ as discussed next.

For an entry $e_i = \langle l_i, m_i, v_i, p_i \rangle$ of the memory from Figure 1, the subset of tests applied to the circuit consists of the test $t_{i,l,m}$ for every pair of values $1 \leq l \leq l_i$ and $1 \leq m \leq m_i$. The cyclic test $t_{i,l,m}$ is applied as follows.

A random initial state is scanned in to ensure that the state of the circuit is known before applying the test. This is important for initializing the circuit. The generation of a random initial state requires an $LFSR$ to drive the scan chains. Starting from the second test, instead of a random initial state, it is possible to use the final state of the previous test.

After initialization, the sequence SE_m is applied to the scan enable input, the sequence $SI_{i,l}$ is applied to the scan chain inputs, and v_i is applied to the primary inputs. The values of the primary outputs and scan chain outputs are captured in a $MISR$.

The number of applied tests is $\sum\{l_i \cdot m_i : c_i \in C_{tg}\}$. The number of clock cycles for the application of every test is $3K + 1$ (K clock cycles for scanning in a random initial state, and $2K+1$ clock cycles for a cyclic test). For L tests, the number of clock cycles is $L(3K + 1) \approx 3LK$. This can be reduced to $L(2K + 1) \approx 2LK$ without the random initialization. For a conventional scan-based test set with L tests, the number of clock cycles is $K + L(K + 1) \approx LK$.

III. SOFTWARE PROCEDURE FOR SELECTING TEST DATA

The software procedure described in this section accepts a scan-based test set C, and a set of target faults F detected by C. The procedure selects a subset $C_{tg} \subseteq C$ for which test data will be stored on-chip, and the parameters l_i and m_i for every test $c_i \in C_{tg}$.

The procedure has two subprocedures described next. Both subprocedures consider lower bounds l_{min} and m_{min} on l and m, respectively. This is important since small values of l and m are not likely to produce effective cyclic tests.

A. Forming C_{tg}

The first subprocedure forms a subset $C_{tg} \subseteq C$, and selects an entry $e_i = \langle l_i, m_i, v_i, p_i \rangle$ for every test $c_i \in C_{tg}$. The goal of the first subprocedure is to ensure that $l_{max} = max\{l_i : c_i \in C_{tg}\}$ and $m_{max} = max\{m_i : c_i \in C_{tg}\}$ are as small as possible to control the storage requirements and the number of applied tests.

The first subprocedure also forms a cyclic test set denoted by T_{tg}. The tests in T_{tg} are the ones based on C_{tg} that are effective in detecting target faults. The set T_{tg} is not stored on-chip. It is used only by the software procedure to keep track of cyclic tests that increase the fault coverage. Initially, $C_{tg} = \emptyset$ and $T_{tg} = \emptyset$.

The procedure considers values of l and m in an order based on their product. Let the product be $q = l \cdot m$. The procedure considers $q = l_{min} \cdot m_{min}$, $l_{min} \cdot m_{min} + 1$, $l_{min} \cdot m_{min} + 2$, ..., $K \cdot (K+1)$, $l = l_{min}$, $l_{min} + 1$, ..., K, and $m = m_{min}$, $m_{min} + 1$, ..., $K + 1$, such that $l \cdot m = q$.

For every combination of q, l and m, the procedure considers every test $c_i \in C$. Based on l, m and c_i, it forms the cyclic test $t_{i,l,m}$. It performs fault simulation with fault dropping of F under $t_{i,l,m}$. If the test detects any faults, it is added to T_{tg}.

After T_{tg} detects all the faults in F, the procedure performs forward-looking reverse order fault simulation to remove unnecessary tests from T_{tg}. Considering the tests that remain in T_{tg}, the procedure constructs C_{tg}, and the entry e_i for every $c_i \in C_{tg}$, as follows.

A test c_i is included in C_{tg} only if T_{tg} contains a test of the form $t_{i,l,m}$. Considering all the tests of the form $t_{i,l,m}$ in T_{tg}, the maximum value of l determines l_i. The maximum value of m determines m_i. Thus, $C_{tg} = \{c_i \in C : t_{i,l,m} \in T_{tg}\ for\ some\ l\ and\ m\}$, $l_i = max\{l : t_{i,l,m} \in T_{tg}\}$, and $m_i = max\{m : t_{i,l,m} \in T_{tg}\}$.

B. Reducing the Number of Tests in C_{tg}

The goal of the second subprocedure is to reduce the number of tests in C_{tg} without increasing l_{max} or m_{max}. A secondary objective is to reduce the values of l_i and m_i when possible. The second subprocedure uses the tests in C_{tg} to construct a new test set T_{tg}. Initially, $T_{tg} = \emptyset$. The procedure considers the tests from C_{tg} one by one from high to low value of $l_i \cdot m_i$. For a test $c_i \in C_{tg}$ with $e_i = \langle l_i, m_i, v_i, p_i \rangle$, the procedure considers all the tests based on e_i. Thus, for $l = l_{min}$, $l_{min} + 1$, ..., l_i, and for $m = m_{min}$, $m_{min} + 1$, ..., m_i, the procedure forms the cyclic test $t_{i,l,m}$. It performs fault simulation with fault dropping of F under $t_{i,l,m}$. If the test detects any faults, it is added to T_{tg}.

Similar to the first subprocedure, the second subprocedure performs forward-looking reverse order fault simulation to remove unnecessary tests from T_{tg}. Considering the tests that remain in T_{tg}, the procedure reconstructs C_{tg}, and the entry e_i for every $c_i \in C_{tg}$.

The second subprocedure simulates all the cyclic tests based on a scan-based test $c_i \in C_{tg}$ before another test from C_{tg} is considered. In addition, scan-based tests with lower values of l_i and m_i are less likely to contribute cyclic tests to T_{tg}. When a test $c_i \in C_{tg}$ does not contribute any cyclic tests to T_{tg}, it is excluded from C_{tg}. If a test c_i with lower values of l_i and m_i is not removed from C_{tg}, the values of l_i and m_i may be reduced to reduce the storage requirements and the number of cyclic tests that will be applied.

IV. Experimental Results

The results of the software procedure for benchmark circuits that can be simulated under cyclic tests using available academic tools are presented in this section.

The scan-based test set C is a compact test set for single stuck-at faults. The set of target faults F consists of single stuck-at faults that are detected by C.

Experimental results indicate that $l_{min} = 1$ and $m_{min} = 1$ result in the lowest number of applied tests and the highest storage requirements. Increasing l_{min} and m_{min} increases the number of tests and decreases the storage requirements. For a circuit with K flip-flops, $l_{min} = K/16$ and $m_{min} = K/16$ balance the number of tests and storage requirements. The value of $K/16$ is increased to the next power of two. This value is denoted by Q_{MIN}.

For most of the circuits considered, the first subprocedure is applied only with $l_{min} = m_{min} = Q_{MIN}$. The second subprocedure is applied several times with $l_{min} = m_{min} = Q_{MIN}$, $Q_{MIN}/2$, ..., 1. As l_{min} and m_{min} are decreased, each scan-based test in C_{tg} is utilized for applying additional cyclic tests, and the number of scan-based tests that remain in C_{tg} is decreased. For several circuits, both subprocedures are also applied with $l_{min} = m_{min} = 1$ to illustrate the possibility of reducing the number of applied tests. The name of the circuit is followed by ".1" in this case.

For comparison, the approach from [15] was applied to single stuck-at faults using the same scan-based test set C. The approach from [15] stores deterministic scan vectors from C in multiple subsets. It uses each subset to produce a fixed number of scan-based tests by selecting scan vectors randomly from the subset for each test. The number of subsets, the number of scan vectors in each subset, and the number of scan-based tests applied for every subset, are increased gradually to find a solution that minimizes the storage requirements and the number of applied tests, while achieving complete fault coverage.

In Table V, after the circuit name, column [15] describes the results of the approach from [15]. Subcolumn $bits$ shows the number of storage bits. Subcolumn app shows the number of tests that need to be applied to achieve complete single stuck-at fault coverage. Column $cyclic$ describes the results using cyclic tests as suggested in this article. Subcolumn $bits$ shows the number of storage bits. Subcolumn $inc(x)$ shows the increase in the number of bits relative to [15]. Subcolumn app shows the number of tests that need to be applied to achieve complete single stuck-at fault coverage. Subcolumn $red(x)$ shows the reduction in the number of applied tests relative to [15]. A reduction higher than 2x (or 3x with random initialization) implies a reduction in the number of clock cycles required for test application.

For further comparison, the number of bits for two additional approaches are shown in Table V. Both approaches apply conventional scan-based tests and require small numbers of tests. The approach from [22] stores scan vectors as well as permutations of scan vectors that are needed for achieving complete fault coverage. In [23], $LFSR$ seeds are computed for the application of a deterministic test set. Each seed is used for producing several tests by complementing bits of the seed and using the same seed with different $LFSR$s. This approach is closer to the approaches described in earlier works.

TABLE V
Comparison

	[15]		cyclic				[22]	[23]
circuit	bits	app	bits	inc(x)	app	red(x)	bits	bits
systemcdes	216	768	1755	8.12	19	40.42	3762	672
sasc	144	384	109	0.76	53	7.25	1104	310
des_area	720	1152	26182	36.36	283	4.07	4320	990
b05	120	20480	204	1.70	284	72.11	1188	-
s1423	320	8192	326	1.02	410	19.98	1380	684
b04	108	49152	359	3.32	524	93.80	1305	636
usb_phy	176	1024	156	0.89	536	1.91	1100	445
i2c	624	24576	756	1.21	920	26.71	4464	1497
spi	1088	32768	2203	2.02	1882	17.41	11322	5300
simple_spi	364	28672	655	1.80	1886	15.20	2580	1204
b07	192	98304	236	1.23	4227	23.26	1162	819
systemcaes	992	4096	2751	2.77	5783	0.71	16833	1811
s5378	1920	16384	2272	1.18	27897	0.59	4998	2840
s13207	5616	53248	1481	0.26	31050	1.71	4224	6147
pci_spoci_ctrl	1040	425984	3477	3.34	84570	5.04	6066	5237
sasc.1	144	384	240	1.67	24	16.00	1104	310
usb_phy.1	176	1024	292	1.66	117	8.75	1100	445
s5378.1	1920	16384	2733	1.42	15140	1.08	4998	2840

Detailed results using cyclic tests are given in Table VI. There are two rows for every circuit in Table VI, corresponding to the first and second subprocedures. For the second subprocedure, the results are shown for the lowest values of l_{min} and m_{min} that reduced the storage requirements or number of applied tests.

After the circuit name, column sv of Table VI shows the number of state variables. Column pi shows the number of primary inputs. Column C shows the number of tests in C. Column pr shows the subprocedure applied. Columns l_{min}, l_{max} and m_{max} show the values of the corresponding parameters. Column C_{tg} shows the number of tests in C_{tg}. Column $bits$ shows the number of storage bits. Column $frac$ shows the number of storage bits as a fraction of the number of bits required for storing C. Column app shows the number of tests that need to be applied. The circuits in Table VI are ordered from low to high number of applied tests. Column $f.c.$ shows the single stuck-at fault coverage. Column $ntime$ shows the normalized runtime for each subprocedure separately. This is the total runtime for the subprocedure, divided by the runtime for fault simulation of cyclic tests based on C.

The following points can be seen from Tables V and VI. The parameters selected by the software procedure result in storage requirements that are similar to the ones produced by the procedure from [15]. Compared with the scan-based test set C, the number of bits is reduced significantly. The number of bits is typically also reduced compared with [22] and [23].

With cyclic tests, the number of applied tests is typically smaller than in [15]. This is made possible by using scan enable sequences with more functional capture cycles.

Whereas the first subprocedure is important for limiting the values of l_{max} and m_{max}, the second subprocedure is important for reducing the number of stored entries, the storage requirements and the number of applied tests.

The normalized runtime of the software procedure is similar for circuits of different sizes. Thus, the procedure scales similar to a fault simulation procedure for cyclic tests. With an efficient fault simulation tool, the procedure is expected to be applicable to larger circuits than the ones in Table VI. The normalized runtime is a stronger function of the difficulty of testing the circuit than its size. The difficulty is higher for higher values of l_{max}, m_{max}, and number of applied tests.

V. Concluding Remarks

This article described a storage-based $LBIST$ approach that uses a unique type of scan-based tests referred to as cyclic tests, with

TABLE VI
EXPERIMENTAL RESULTS

circuit	sv	pi	C	pr	l_{min}	l_{max}	m_{max}	C_{tg}	bits	frac	app	f.c.	ntime
systemcdes	190	130	79	1	16	16	17	16	2496	0.099	4112	100.000	0.86
systemcdes	190	130	79	2	1	1	5	13	1755	0.069	19	100.000	7.03
sasc	117	15	22	1	8	8	9	11	341	0.117	744	100.000	2.70
sasc	117	15	22	2	1	3	7	5	109	0.038	53	100.000	15.90
des_area	128	239	118	1	8	8	62	106	27242	0.629	8024	100.000	83.52
des_area	128	239	118	2	1	1	62	106	26182	0.605	283	100.000	144.04
b05	34	2	61	1	2	8	12	23	307	0.140	384	95.209	15.71
b05	34	2	61	2	1	8	12	15	204	0.093	284	95.209	4.79
s1423	74	17	26	1	4	13	34	15	496	0.210	737	99.076	50.49
s1423	74	17	26	2	1	13	34	10	326	0.138	410	99.076	44.58
b04	66	12	44	1	4	18	18	20	544	0.159	1015	99.851	32.44
b04	66	12	44	2	1	18	16	14	359	0.105	524	99.851	50.11
usb_phy	98	14	32	1	8	13	18	19	618	0.172	1963	100.000	31.48
usb_phy	98	14	32	2	1	12	18	5	156	0.044	536	100.000	27.58
i2c	128	17	45	1	8	31	60	31	1181	0.181	3612	100.000	83.71
i2c	128	17	45	2	1	31	60	24	756	0.116	920	100.000	142.01
spi	229	45	406	1	16	41	30	99	7207	0.065	28080	99.985	20.97
spi	229	45	406	2	1	41	21	37	2203	0.020	1882	99.985	50.33
simple_spi	131	15	36	1	8	122	13	24	1141	0.217	5984	100.000	90.59
simple_spi	131	15	36	2	1	122	13	18	655	0.125	1886	100.000	331.07
b07	51	2	52	1	4	33	52	21	495	0.180	5016	99.915	128.47
b07	51	2	52	2	1	33	52	8	236	0.086	4227	99.915	187.53
systemcaes	670	258	121	1	64	70	66	44	14798	0.132	181700	99.995	6.52
systemcaes	670	258	121	2	16	47	60	8	2320	0.021	5783	99.995	199.37
s5378	179	35	100	1	16	76	128	69	4761	0.222	54857	99.131	363.65
s5378	179	35	100	2	1	76	128	38	2272	0.106	27897	99.131	3089.47
s13207	669	31	235	1	64	80	98	103	11340	0.069	459300	98.462	53.19
s13207	669	31	235	2	8	80	98	17	1212	0.007	31050	98.462	73.91
pci_spoci_ctrl	60	23	146	1	4	54	61	78	5210	0.430	95844	99.942	3970.07
pci_spoci_ctrl	60	23	146	2	2	54	61	47	3477	0.287	84570	99.942	888.01

cyclic scan enable and scan in sequences. Cyclic tests provide improved fault detection capabilities since their scan enable sequences have higher proportions of functional capture cycles compared with conventional scan-based tests. Experimental results for benchmark circuits demonstrated that the improved fault detection capabilities help balance the number of applied tests and the storage requirements.

REFERENCES

[1] P. H. Bardell, W. H. McAnney and J. Savir, $Built-In\ Test\ for\ VLSI\ Pseudorandom\ Techniques$, Wiley Interscience, 1987.

[2] N. A. Touba and E. J. McCluskey, "Bit-fixing in Pseudorandom Sequences for Scan BIST", in IEEE Trans. on Computer-Aided Design, April 2001, Vol. 20, No. 4, pp. 545-555.

[3] S. Hellebrand, H.-G. Liang and H.-J. Wunderlich, "A Mixed Mode BIST Scheme Based on Reseeding of Folding Counters", Journal of Electronic Testing, 2001, Vol. 17, pp. 341–349.

[4] I. Pomeranz and S. M. Reddy, "A Storage Based Built-In Test Pattern Generation Method for Scan Circuits Based on Partitioning and Reduction of a Precomputed Test Set", in IEEE Trans. on Computers, Nov. 2002, pp. 1282-1993.

[5] S. Pateras, "Security vs. Test Quality: Fully Embedded Test Approaches are the Key to Having Both", in Proc. Intl. Test Conf., 2004, Panel P2.2, p. 1413.

[6] D. Xiang, M. Chen and H. Fujiwara, "Using Weighted Scan Enable Signals to Improve Test Effectiveness of Scan-Based BIST", in IEEE Trans. on Computers, Dec. 2007, vol. 56, no. 12, pp. 1619-1628.

[7] L.-T. Wang, X. Wen, S. Wu, H. Furukawa, H.-J. Chao, B. Sheu, J. Guo and W.-B. Jone, "Using Launch-on-Capture for Testing BIST Designs Containing Synchronous and Asynchronous Clock Domains", in IEEE Trans. on Computer-Aided Design, Feb. 2010, Vol. 29, No. 2, pp. 299-312.

[8] R. S. Oliveira, J. Semiao, I. C. Teixeira, M. B. Santos and J. P. Teixeira, "On-line BIST for Performance Failure Prediction under Aging Effects in Automotive Safety-critical Applications", in Proc. Latin American Test Workshop, 2011, pp. 1-6.

[9] Y. Sato, H. Yamaguchi, M. Matsuzono and S. Kajihara, "Multi-Cycle Test with Partial Observation on Scan-Based BIST Structure", in Proc. Asian Test Symp., 2011, pp. 54-59.

[10] M. E. Imhof and H. Wunderlich, "Bit-Flipping Scan - A Unified Architecture for Fault Tolerance and Offline Test", in Proc. Design, Automation & Test in Europe Conf., 2014, pp. 1-6.

[11] C. Shiao, W. Lien and K. Lee, "A Test-per-cycle BIST Architecture with Low Area Overhead and no Storage Requirement", in Proc. Intl. Symp. on VLSI Design, Automation and Test, 2016, pp. 1-4.

[12] B. Kaczmarek, G. Mrugalski, N. Mukherjee, J. Rajski, Ł. Rybak and J. Tyszer, "Test Sequence-Optimized BIST for Automotive Applications", in Proc. European Test Symp., 2020, pp. 1-6.

[13] A. Koneru and K. Chakrabarty, "An Interlayer Interconnect BIST and Diagnosis Solution for Monolithic 3-D ICs", in IEEE Trans. on Computer-Aided Design, Oct. 2020, Vol. 39, No. 10, pp. 3056-3066.

[14] I. Pomeranz, "Storage-Based Built-In Self-Test for Gate-Exhaustive Faults", in IEEE Trans. on Computer-Aided Design, Oct. 2021, Vol. 40, No. 10, pp. 2189-2193.

[15] I. Pomeranz, "Zoom-In Feature for Storage-Based Logic Built-In Self-Test", in Proc. Intl. Symp. on Defect and Fault Tolerance in VLSI and Nanotechnology Systems, 2021.

[16] M. B. G. Remadevi and R. Bakthavatchalu, "Design of a Programmable Low Power Linear Feedback Shift Register for BIST Applications", in Proc. Intl. Test Conf. India, 2022, pp. 1-4

[17] F. Zhang, D. Hwong, Y. Sun, A. Garcia, S. Alhelaly, G. Shofner, L. Winemberg and J. Dworak, "Putting Wasted Clock Cycles to Use: Enhancing Fortuitous Cell-aware Fault Detection with Scan Shift Capture", in Proc. Intl. Test Conf., 2016 pp. 1-10.

[18] G. Mrugalski, J. Rajski, J. Solecki, J. Tyszer and C. Wang, "Trimodal Scan-Based Test Paradigm", IEEE Trans. on VLSI Systems, March 2017, Vol. 25, No. 3, pp. 1112-1125.

[19] Y. Liu, J. Rajski, S. M. Reddy, J. Solecki and J. Tyszer, "Staggered ATPG with Capture-per-cycle Observation Test Points", in Proc. VLSI Test Symp., 2018, pp. 1-6.

[20] I. Pomeranz and S. M. Reddy, "Transparent Scan: A New Approach to Test Generation and Test Compaction for Scan Circuits that Incorporates Limited Scan Operations", IEEE Trans. on Computer-Aided Design, Dec. 2003, pp. 1663-1670.

[21] X. Lin, R. Press, J. Rajski, P. Reuter, T. Rinderknecht, B. Swanson and N. Tamarapalli, "High-frequency, At-speed Scan Testing", in IEEE Design & Test of Computers, Sept.-Oct. 2003, vol. 20, no. 5, pp. 17-25.

[22] S. Gopalsami and I. Pomeranz, "Fully Deterministic Storage Based Logic Built-In Self-Test", Technical Report.

[23] I. Pomeranz, "Input Test Data Volume Reduction Using Seed Complementation and Multiple $LFSR$s", in Proc. VLSI Test Symp., 2020.