Journal of Electronic Testing (2022) 38:107–123
https://doi.org/10.1007/s10836-022-05986-0

A Low-cost BIST Design Supporting Offline and Online Tests

Ahmad Menbari[1] · **Hadi Jahanirad**[1]

Received: 29 November 2021 / Accepted: 14 February 2022 / Published online: 10 March 2022

Abstract
Offline and online built-in self-test (BIST) designs are low-cost platforms to test very complex modern chips. The offline BIST design embeds the test pattern generator (TPG) into the chip to be activated in the test time. On the other hand, the online (or concurrent) BIST design eliminates the TPG and utilizes the system's input vectors to accomplish the test process. This paper proposes a BIST design that supports both online and offline tests. In the online part of the design, a selector module passes the input vectors which belong to a pre-computed test set to the reduction part. The test set contains the test vectors, which generate 0 remainders in the division by the LFSR's polynomial of the selector. In the concurrent test latency (CTL) aware design, the size of the test set is expanded by adopting the selecting part to select the test vectors which generate the same remainders in the division by two different polynomials. The internal TPG of the offline part is realized based on the HW-aware test set using the shifted versions of LFSR's polynomial and XORing their contents. The reduction part compresses the widths of the current test vector and the related CUT outputs. The compactor LFSR compresses the test vectors so that the resulted remainders would be different for all test vectors to solve the masking problem. The small size of the test set and the compacting test vectors resulted in a tremendous reduction of hardware overhead. The proposed method imposes less than 6% and 28% hardware overhead for large size and very large size circuits, respectively. The simulation results for ISCAS 85, ISCAS 89, and ITC99 benchmark circuits showed that our proposed BIST design outperforms the previous state-of-the-art in both hardware overheads. Furthermore, the CTL reduces 100 times by the proposed CTL-aware approach on average.

Keywords Built-in self-test · Concurrent self-test · Concurrent test latency · Linear feedback shift register

1 Introduction

With the advent of Very Large System Integration (VLSI), the fault occurrence increase in modern chips [9, 22]. The Built-in self-test (BIST) is an efficient technique to detect and diagnose such faults. The main advantage of BIST design is embedding the required test modules into the chip, which eliminates the inter-chip communications and improves the system's reliability [1]. BIST reduces the test process cost and allows testing the main circuit independent of high cost and low-speed automatic test generator (ATG) equipment [28]. Economic benefits of BIST results in developing BIST platforms for mixed-signal chips such as data converters (ADC and DAC) and system on chip platforms (SoCs) [7, 17, 18].

Built-in self-test (BIST) approaches are classified into **offline** and **online** categories [1]. In the offline-BIST design, the regular operation of the circuit under test (CUT) stalls, then a test pattern generator (TPG) applies test vectors to the CUT's inputs. An output response analyzer (ORA) decides every test vector's pass/fail state. After completing the test process, the circuit returns to normal mode. The TPG of offline BIST design is implemented by one of the exhaustive, pseudo-random, and deterministic approaches [2, 10, 14]. Developing efficient offline-BIST design is an active area of research; for example, several low power (LP) BIST methods have been developed recently in [3, 15, 24].

Two main drawbacks of offline BIST are as follows. First, the offline approach cannot detect temporary faults that frequently occur in modern VLSI chips. Second, the regular operation should stall to perform the offline test that degrades the performance of the CUT.

Responsible Editor: K. K. Saluja

✉ Hadi Jahanirad
h.jahanirad@uok.ac.ir

[1] Department of Electrical Engineering, University of Kurdistan, 80523 Sanandaj, Kurdestan, Iran

The online BIST approach solves the above problems utilizing the applied input vectors to test the CUT. The test operation proceeds in CUT's normal mode, so the circuit doesn't stall. Furthermore, the continuous test process of the online BIST approach would increase the detection probability of intermittent and transient faults [6, 8, 12, 31].

The online BIST design employs a selecting part to select the input vectors which belong to a pre-computed test set [11]. Generally, the pre-computed test vectors are stored in a memory, and then the incoming input vector is compared with one or multiple test vectors. If the input vector matches the related test vector, then the output vector of CUT is compared with the expected vector to determine CUT's fail/pass state.

A significant drawback of the online BIST approach is its long test time completion. Moreover, the hardware overhead of the selection part may be very high. Various online BIST designs have been presented to handle these two problems. Some deal with the hardware overhead, while others simultaneously test time and hardware overhead. The following section will introduce a brief review of the previous designs.

This paper proposes a BIST design that can perform both offline and online tests. Switching between offline and online test modes is on user demand. Whether the current input vector belongs to the pre-computed test set or not is determined using an LFSR. So, the memory overhead regarding the pre-computed test set registration is eliminated. Moreover, there is no need to compare the input vector with the active test vector. In the next step, the selected input vector and the related output vector are compressed based on two different LFSRs. The chosen input vector is applied to a compact version of CUT (called golden circuit (GC)), and finally, a comparator compares the output of the golden circuit with the compacted version of CUT's output to decide the fail/pass state of the system. A low-cost TPG is employed to test the CUT when offline test mode is selected. A multiplexer connects the TPG's output to the CUT's inputs in this mode. The offline test is completed in a short time. Then the system returns to the online mode by connecting the CUT's inputs to the primary inputs using the multiplexer.

The rest of the paper is as follows: in Sect. 2, the related works are described. Due to the critical role of the LFSR in the proposed method, the associated preliminaries are presented in Sect. 3. In Sect. 4, the proposed BIST design is presented. Then its application to a case study will be reported in Sect. 5. The comparison results with some of the previous methods are reported in Sect. 6. Finally, the paper is concluded in Sect. 7.

2 Related Work

Hardware overhead and CTL (the time required to complete the test in online mode) are two critical metrics for assessing the concurrent BIST designs. Therefore, we evaluate the state-of-the-art methods according to these metrics.

The most straightforward online BIST architecture is duplication design, wherein the input vector is applied to the CUT and its copy. A comparator checks the equality of two circuits' output vectors for every applied input vector. The CTL of duplication design is minimum because all system input vectors belong to the test set. However, its hardware overhead is more than 100% (CUT's copy and the output verifier).

Another viewpoint of online BIST is to use a pre-determined test set. The test vectors in the pre-computed test set are selected so that a pre-requisite fault coverage is guaranteed [20, 21, 27]. If the number of circuit's input be small, then the test set contains all input vectors (exhaustive test). Composition of test set can be done using a test generation algorithm, or the test vectors may be selected randomly [29, 30].

CBIST [22] is one of the early concurrent BIST designs. A test vector from the test set is selected as the active test vector, and selecting part waits until it occurs in the circuit's primary inputs. When the comparison of the output vector with the expected output vector is completed, the fail/pass state of the circuit is determined. The next test vector from the test set is considered the active test vector. The above process is repeated until all test vectors are checked. The main drawback of CBIST is high concurrent test latency (CTL) due to the unpredictable wait time of matching the active test vector with the circuit's input vector.

Multiple Hardware Signature Analysis Technique (MHSAT) [20], *Order Independent Signature Analysis Technique* (OISAT) [21], *windowed-ComparativeConcurrent BIST* (WCBIST) [27], and *RAM-based Input Vector Monitoring Concurrent BIST* (R-CBIST) [29] try to alleviate the high CTL of CBIST through increasing the number of active test vectors. The probability of matching CUT's input vector and multiple active test vectors is much higher than matching with only one active test vector. It is worth noting that the test set of all the abovementioned techniques is generated based on pseudo-random algorithms. In large CUTs, the test set would be large, resulting in impractical CTL.

In MHSAT [20], the active test set consists of L active test vectors, which L LFSRs produce. A hit occurs when the input vector matches any active test vector. The related

LFSR proceeds to the next state, and the corresponding response verifier implemented by Multiple Input Shift Registers (MISR) is activated. When all LFSRs sweep their states the test would be completed. OISAT [21] takes similar L LFSR for active test vector generation but utilizes the Accumulator-Based Compaction (ABC) as a response verifier.

The w-CBIST [27] divides all possible input vectors into windows containing W vectors. In each step, a window is considered an active window, and the matching of the current input vector with the vectors belonging to the test vectors in the active window is examined. The test will be completed when all windows are processed.

Some concurrent BIST designs are based on a pre-computed test set in which the test vectors are selected by deterministic TPG algorithm [5, 13, 16, 19, 26–28]. These techniques decrease the size of the test set, and the CTL reduces to some extent.

BICST [23] consists of a concurrent test circuit (CTC) as a pattern detector implemented by a programmable logic array (PLA). The number of inputs (n) and outputs (m) of CTC and CUT is equal. Whenever a test vector occurs in the CUT's input port, the comparator (output verifier) makes the *pass/fail* decision by comparing the CUT and CTC outputs.

MICSET [28] architecture improves the BICST design by adding an offline test mechanism and modification of the pattern detector. Suppose that the test set contains T n-bit test vectors. First, a $T \times n$ test matrix is generated and based on a greedy algorithm, t ($\log_2 T < t < T$) different columns of the test matrix are selected such that all T t-tuples are distinct. These t columns direct the remaining (n-t) bits of test vectors constructed by the OR-plane module. In the offline mode, a set of 2×1 multiplexers connect the generated test vectors by the t-stage counter and (n-t)-bits from an OR-plane to the CUT inputs. Consequently, the MICSET would support both online and offline test modes.

Both BICST and MICSET have a high hardware overhead. Furthermore, due to the extra low probability of pre-computed test vectors occurrence in each clock cycle, the CTL is impractical for circuits with more than 40 primary inputs [28]. The authors of [26] use SRAM cells to monitor test vectors. This design's hardware overhead and CTL are more efficient than the previous input vector monitoring approaches.

NEMO [25] and later a cost-efficient NEMO [32] proposed to reduce the area overhead. Figure 1 illustrates the schematic of NEMO. The number of pre-computed test vectors is T ($T << 2^n$), which are detected by a decoding module (D). The multi-level decoder is implemented by 2-input AND gates wherein if one of the test vectors occurs in the input of CUT; the related output is set to 1. The CALC module generates the compacted versions of related CUT outputs to reduce the hardware overhead. On the other side, a space compactor (SC) compresses the width of the CUT output vector from m to q bits. These compacted versions of CALC and SC modules are compared and, the fail/pass state of the system is reported. The output of the OR module equals 1 only when one of the T outputs of the decoder is 1 (which means a test vector occurs in the primary inputs).

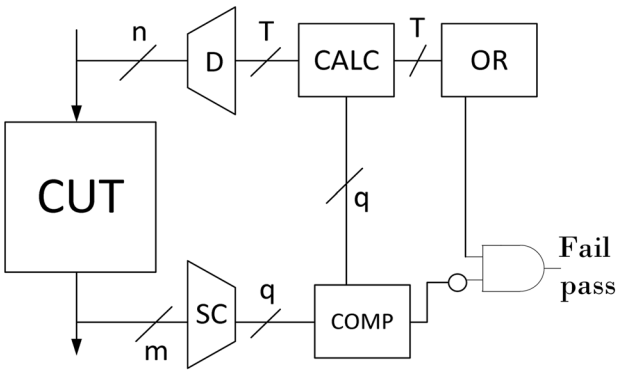

Fig. 1 NEMO schematic

NEMO's space compactor (SC) is realized using the methodology represented in [4], which requires the CUT's details. So, if these details are not available, constructing SC would be infeasible. This problem has been solved in cost-efficient NEMO (CE-NEMO) utilizing another decoder instead of SC in the output of CUT. Furthermore, three modifications in the input side decoder have been applied, which result in more hardware overhead reduction:

1. A more efficient approach is applied to reduce the columns of the pre-computed test set
2. When the number of distinct columns is not a power of 2, the self pairing of unpaired columns is eliminated, and such columns are directly passed to the next decoding level.
3. The most efficient ordering of distinct columns is derived using a meta-hueristic optimization approach based on Simulated Annealing (SA).

As a problem, the hardware complexity of decoder modules in both NEMO and CE-NEMO is highly dependent on the number of pre-computed test vectors. Increasing the size of the test set would significantly increase the number of AND gates in multilevel decoders. Moreover, the bit-width of input vectors affects both hardware overhead and design complexity of decoders. Due to the SA-based optimization approach to finding the most efficient order of inputs, the severity of this problem is increased for CE-NEMO. On the other hand, NEMO and CE-NEMO minimize hardware overhead that eventually results in very high concurrent test latency.

[13] proposed a method with low CTL and acceptable hardware overhead. This method (DC-based) is based on the idea that if the input port of CUT contains n pins, then there are t < n pins that cover many CUT faults. The other n-t pins are *don't care* bits. Consequently, for every combination of t bits, 2n-t input vectors would be recognized as test vectors. Therefore, the CTL reduces significantly as well as the existence of don't care bits leads to more simplification in the pattern detector circuit. In the DC-based method, all the required modules to perform testing (e., g., the pattern detector and output verifier) are synthesized as a logic module.

Similar to NEMO and CE-NEMO, the hardware overhead of the DC-based method is highly dependent on the number and bit-width of pre-determined test vectors. Furthermore, the maximum fault coverage would be reduced by reducing the width of specified bits, so there is a compromise between the hardware overhead reduction and achievable fault coverage. On the other hand, if the number of test vectors increases to reduce CTL, the hardware overhead would increase accordingly.

To overcome the above mentioned issues, we propose a BIST design wherein both online and offline test process are supported. These two parts are totally independent so the offline part can be removed to achieve more hardware saving.

The selecting module of the online part is constructed based on an LFSR. The characteristic polynomial of LFSR is defined to generate zero-remainder for input vectors belonging to the pre-computed test set. Consequently, the complex decoders of NEMO, CE-NEMO, and DC-based designs are replaced with just an LFSR. The other advantage of the proposed selecting part is that by changing the test set, the length and polynomial of the LFSR would change, and there is no need to re-design the selecting part architecture.

To test the correctness of CUT's output vector for pre-determined test vectors, we compact both CUT's input and output vectors using two LFSRs. The compressed version of the input vector is applied to a golden circuit which generates the correct compressed version of related CUT's output. The proposed approach significantly reduces the design complexity of compaction and comparison modules of NEMO, CE-NEMO, and DC-based designs.

On the other hand, a CTL-aware approach would be presented, which achieves a tremendous reduction of test completion time by expanding the pre-computed test set. In summary, the contributions of the paper are as follows:

1) Developing a low hardware overhead BIST design that supports online and offline test modes.
2) Developing an LFSR-based approach for constructing a pre-computed test set for specific fault coverage.
3) Developing an LFSR-based test set selection with low concurrent test latency.
4) Presentation of a low hardware overhead and low test-time offline test pattern generator.

3 Preliminaries

Because of the critical role of LFSR in the proposed method, the main related concepts are described in this section.

A polynomial in Galois Field algebra represents a binary number (e.g., byte, word, etc.). An N-bit binary number is converted to an algebraic polynomial of order N-1. For instance, the eight-bit (a byte) binary number of Sbin would be converted to a polynomial with order 7 according to (1).

$$S_{bin} = \left(b_7 b_6 b_7 b_6 b_7 b_6 b_7 b_6\right) => S(x) = b_7 X^7 + b_6 X^6 + \cdots + b_1 X^1 + b_0 \quad (1)$$

The modular-2 Galois field addition and multiplication are equivalent to XOR and AND operations, respectively.

$1+1=0,\ 1+0=1,\ 0+0=0$

$1\times 1=1,\ 1\times 0=0,\ 0\times 0=1$

For example, the addition and multiplication of 110 and 101 are calculated as follows:

$110 => X^2+X$

$101 => X^2+1$

$110+101=X^2+X+X^2+1=011$

$110\times 101=(X^2+X)\times(X^2+1)=11{,}110$

All the LFSRs in our proposed BIST design are Type-2 LFSR (internal XOR LFSR). The structure of Type-2 LFSR is illustrated in Fig. 2, wherein k DFFs are configured in a right-shift register format. Some XOR gates are inserted in the input of DFFs to build the feedback loops. The locations of inserted XOR gates are determined based on the LFSR's characteristic polynomial.

$$L(x) = 1 + C_{k-1}X^{k-1} + \sum_{i=1}^{k-2} C_i X^{k-i} \quad (2)$$

The characteristic polynomial is defined according to (2). The first term indicates that an XOR gate is always available in the rightmost DFF's output. The second term demonstrates that a feedback loop always starts from the leftmost DFF's output and ends on the rightmost DFF's input. The other C_i coefficients will be set to 1 if an XOR

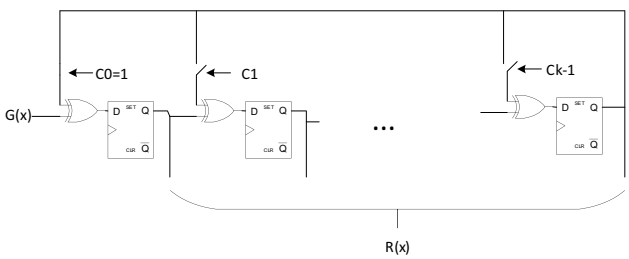

Fig. 2 Type-2 LFSR architecture

Table 1 Arrangement of input vectors' polynomials in the division by L(x)

						R(x)
	0	L(x)	xL(x)	…	f(x)L(x)	0
	1	L(x)+1	xL(x)+1	…	f(x)L(x)+1	1
	x	L(x)+x	xL(x)+x	…	f(x)L(x)+x	x
	x+1	L(x)+x+1	xL(x)+x+1	…	f(x)L(x)+x+1	x+1
G(x) of all primary inputs						
	$x^{k-1}+....+1$	$L(x)+x^{k-1}+....+1$	$xL(x)+x^{k-1}+....+1$	…	$f(x)L(x)+x^{k-1}+....+1$	$x^{k-1}+....+1$

gate exists in the related location. Otherwise, the coefficient would equal 0.

Suppose that an N-bit binary number goes serially through an LFSR with k shift register. If the corresponding polynomial of the N-bit binary number is G(x) and the characteristic polynomial of the LFSR is L(x), then the final content of the LFSR's DFFs would be the remainder of the division of G(x) by L(x). The relationship among G(x), the LFSR's characteristic polynomial (L(x)), and the remainder (R(x)) can be expressed by (3).

$$G(x) = L(x) = Q(x) + R(x) \tag{3}$$

There are 2^N G(x) polynomials of order N. If these polynomials are divided by L(x) with the order of k, the set of remainders would be generated as follows.

$Rem\_Poly = \{0, 1, X, 1+X, X^2, ..., 1+X+X^2+...+X^{k-1}\}$.

Every G(x) would be related to one and only one element of Rem_Poly. In Table 1, all polynomials which generate the same remainder are put in a row. For example, all polynomials of G(x), which create the remainder 0 in the division by L(x), are put in the first row of Table 1.

4 Proposed BIST DESIGN

The overall architecture of the proposed BIST design is indicated in Fig. 3. In the core of the system, the circuit-under-test (CUT). The left part of the system is devoted to offline test mode, and the right part corresponds to the online test mode. The system's upper part consists of a test pattern generator (TPG) module and a multiplexer (MUX_1). Two input channels of MUX_1 are connected to primary inputs and the output of TPG, respectively. A control signal (Tm) defines the online or offline test modes. In the offline test mode ($T_m = 0$), the MUX_1 passes the output of the TPG module to its output. In the online test mode ($T_m = 1$), the MUX_1 gives the primary input vector to its output. Consecutively, the TPG module generates the necessary pre-computed test vectors and applies them to the CUT.

The online part of the BIST design consists of a selecting part, a reduction part, a golden circuit (GC-online), and a comparator (CMP_{online}). The selection part receives the input vector from the MUX_1 and decides if this input vector belongs to CUT's test set. The selecting part utilizes one or two LFSRs for hardware overhead aware and CTL-aware,

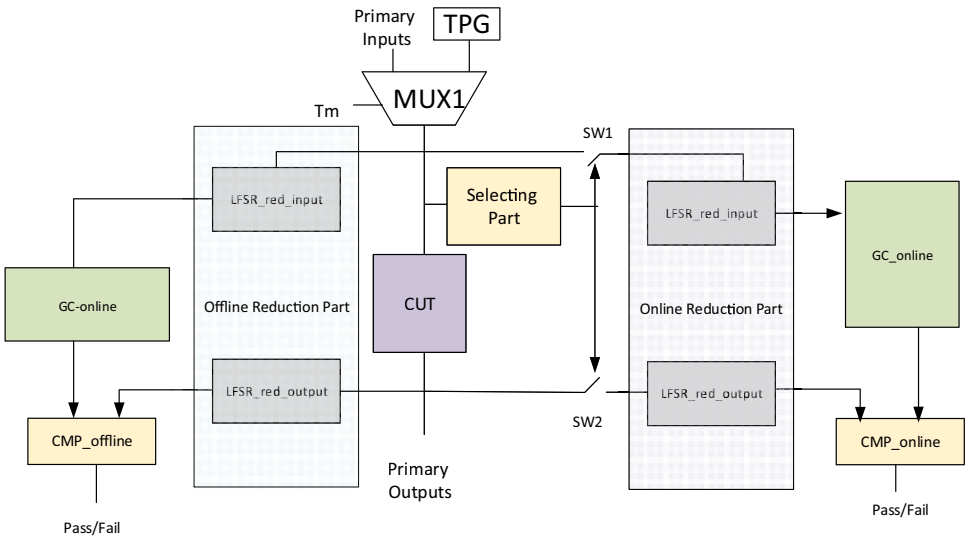

Fig. 3 Schematic of the proposed BIST design

respectively. When one of the test vectors enters the selection part, two switches (SW_1 and SW_2) are closed. So, CUT's input vector and output would be connected to the reduction part. The reduction part compresses the bit widths of the input vector and CUT output vector using two different LFSRs. The size of the GC-online would be significantly reduced due to the mapping of the compressed versions of the input vectors and the corresponding output vectors. The GC-online is synthesized to generate the expected value of the compact version of CUT's outputs for every CUT test vector. Finally, the CMP_{online} declares whether the CUT passes the test process for all test vectors or not.

The offline part consists of the reductions, the offline golden circuit (offlin-GC), and a comparator. The reduction part contains two independent LFSRs, which compact the test vector and CUT's output vector to be applpied to offline-GC and comparator modules, respectively. The offline-GC is synthesized to map the compacted version of test vectors into the compacted version of related CUT's output vectors. Finally, the comparator ($CMP_{offline}$) declares if the generated output of CUT equals the expected value of GC-offline or not.

In the following sub-sections, we describe the details of the proposed BIST design.

4.1 Selecting Part Design

The selecting part passes the pre-computed test vectors toward the GC-online part. The pre-computed test vectors are defined based on single stuck-at fault coverage [1]. Various algorithms have been developed to determine minimum test vectors for achieving required fault coverage. In this paper, we use the LFSR properties to define the pre-computed test vectors as follows.

We propose hardware-aware and CTL-aware approaches for selecting part. In the hardware-aware (HW-aware) approach, we use just the vectors in the first row of Table 1 to test the circuit. In this case, the L(x) should be defined so that the test vectors in the first row can reach the desired fault coverage. In the concurrent test latency aware (CTL-aware) approach, the selected test vectors are defined as the input vectors which generate the same reminders in the division by two different polynomials, $L_1(x)$ and $L_2(x)$. In this case, the test vectors are distributed among the elements of Table 1. So, the concurrent test latency (CTL) reduces compared with the first case.

Utilizing the above approaches, one or two LFSRs substitute the hardware resources (ROM, PLA, and the decoders), which utilize to select the test vectors in previous studies. So, a significant hardware overhead reduction is achieved.

4.1.1 Hardware Overhead Aware Selecting Part

The selecting part selects the input vectors, which generate a 0 remainder in the division by L(x). Consequently, L(x) should be calculated so that the input vectors of the first row of Table 1 would produce the required fault coverage.

For a CUT with N_{inp} (= 64) primary inputs, the total number of input vectors is $2^{N_{inp}} (= 18 \times 10^{18})$. Suppose that the CUT requires at least N_{test} (= 128) vectors to reach the specified fault coverage. Consequently, the number of L(X) candidates is $2^{N_{inp}}/N_{test} (= 144 \times 10^{15})$. For every candidate, a simulator should compute the fault coverage. It would be impractical to check such a tremendous number of options.

To address this problem, we develop the following approach. First, every input test vector should be satisfied (4), wherein f(x) is the coefficient polynomial. The summation of the order of L(x) and f(x) equals the order of G(x). For the L(x) of order *k*, a set of coefficients is defined according to (5). The first coefficient is zero. The last one is the polynomial relates to the most significant order G(x), which is divisible by L(x).

$$G(x) = f(x) \times L(x) \tag{4}$$

$$S_{coeff} = \{0, 1, x, x+1, x^2, x+x^2, 1+x+x^2, \ldots., 1+x+\ldots x^{n-k}\} \tag{5}$$

The optimum L(x) is derived as follows. The proposed algorithm starts with a set of P(x) polynomials derived from a TPG algorithm such as D-Alg. A TPG algorithm selects a set of input vectors based on the circuit's topology and the fault modeling. The goal of the TPG algorithm is to achieve a specified fault coverage. In the ideal case, an L(x) would be found which all of these P(x) polynomials are divisible. If such an L(x) exists, then definitely our LFSR-based selector passes all of the elements of the test set. Elsewhere, we should choose the most suitable one in the maximum number of P(x) polynomials that are divisible, and the necessary fault coverage is achieved.

The following steps are applied to select the most efficient L(x) (Fig. 4):

1. Run D-ALG and extract the required input test vectors ($S_{D\text{-}ALG} = \{P_1, P_2, \ldots, P_{\#test\_vecD\text{-}ALG}\}$) to achieve the specified fault coverage (FC).
2. Based on the number of D-ALG's test vectors (#test_$vec_{D\text{-}ALG}$), construct the L(X)'s coefficient polynomial set. For example, if #test_$vec_{D\text{-}ALG} = 100$, then the smallest set of coefficients would be $S_{Coeff} = \{x, x+1, x^2, \ldots, x^6+x^5+x^2\}$ which contains 100 different coefficients.
3. In the first iteration, for each input vector P_i, we derive the $L_{i,j}(X)$ by dividing $P_i(X)$ by the j'th coefficient in

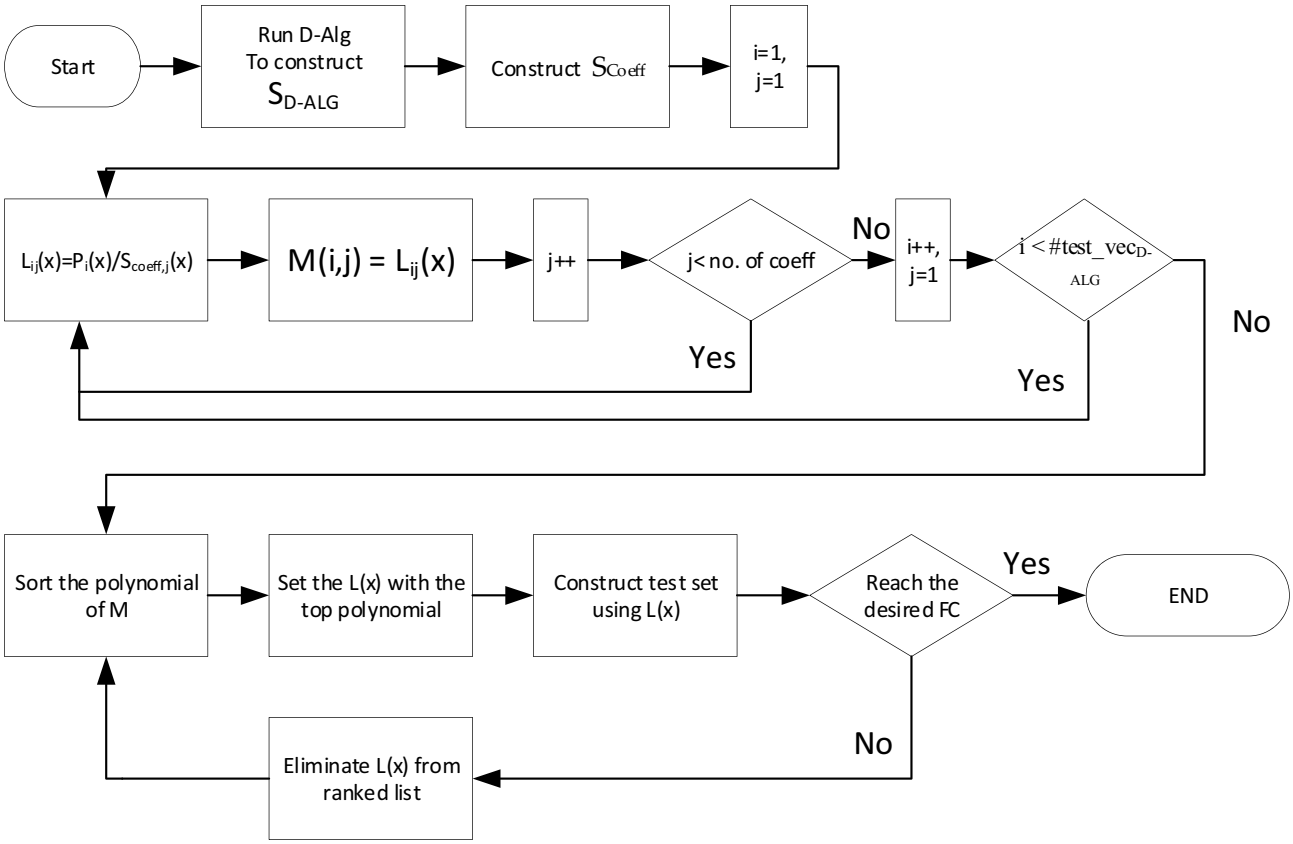

Fig. 4 The flowchart of the most efficient selector polynomial

S_{coeff}. Then save these polynomials in the M_L matrix. A matrix element could be zero if $P_i(x)$ is smaller than the j'th coefficient or the j'th coefficient is not divisible by $P_i(x)$.

4. Determine the different polynomials in the M_L matrix. Then sort these polynomials by the number of repetitions in the M_L matrix.
5. Select the top polynomial of the sorted list as L(X), and construct the first column of Table 1.
6. Run the fault coverage simulator (e.g., ATALANTA) using the generated test vectors in step 5.
7. If desired fault coverage is achieved, the algorithm will be terminated, else add the following polynomial to S_{coeff} and go back to step 2.

In Fig. 5, the number of required test vectors to achieve 90% fault coverage for random TPG, a deterministic TPG algorithm (D-ALG), and the proposed method is presented. For test pattern generation and fault simulation, we use ATALANTA and ModelSim, respectively.

The number of selected test vectors is greater than the number of the test vectors for the deterministic TPG algorithm and is less than the number of the random TPG test vectors. Consequently, as a complex and hardware demanding module in most of the state-of-the-art methods, the pattern detector is replaced by two LFSR's and a comparator (selecting part). As a worth-noting point, the comparator in the HW-aware's selecting part is just a NOR gate where the output is set to 1 if all LFSR bits are set to 0.

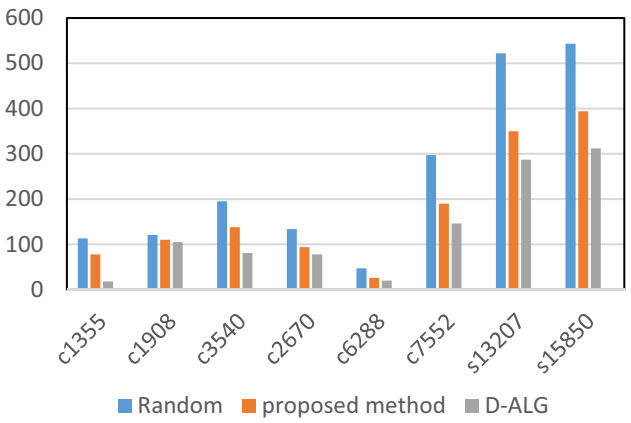

Fig. 5 The number of test vectors for various methods

4.1.2 CTL Overhead Aware Selecting Part

Concurrent test latency (CTL) is the elapsed time for all test vectors occurrence in the primary inputs. there are three assumptions to compute the CTL [22]:

1. An input vector applies to the CUT in every clock cycle.
2. In each clock cycle, all input vectors are likely to occur with the same probability.
3. The occurrence probability for any input vector is independent of the occurrence of the others.

Assume that the number of test vectors to detect the acceptable number of faults is *T*. The probability of occurrence of *i* numbers of test vectors in a clock cycle is defined as the *hit* of *i* test vectors, $h(i) = i/2^n$(n is the number of CUT's inputs). The number of clock cycles required to occur *i* numbers of test vectors is $2^n/i$. Therefore, the CTL is given by

$$CTL = \sum_1^T \frac{1}{h(i)} = 2^n \sum_1^T \frac{1}{i} \tag{6}$$

In the proposed HW-aware method, suppose that the number of test vectors to achieve acceptable fault coverage is *m*. Therefore, the CTL for the proposed HW-aware method is calculated according to Eq. (3)

$$CTL = \sum_1^m \frac{1}{h(i)} = 2^n \sum_1^m \frac{1}{i} \tag{7}$$

As proved by simulation in [28], the CTL has an acceptable value for the circuits containing less than 40 inputs. Otherwise, the CTL would be impractical. In this paper, a CTL-aware concurrent BIST design is proposed. The overall scheme of this design is indicated in Fig. 1. The main differences in comparison with HW-aware design are as follows.

In the selection part, we inserted 2 LFSRs with different primary polynomials, $L_1(x)$ and $L_2(x)$. When an input polynomial G(x) enters these LFSRs, the comparator allows the SW_1 and SW_2 to be connected if both remainders are equal. Consequently, only those polynomials are selected which have the (8) formation.

$$G(x) = m(x)L_1(x)L_2(x) + R(x) \tag{8}$$

Such polynomials generate the same remainder (R(x)) in the division by $L_1(x)$ or $L_2(x)$. the maximum order of R(x) would be the minimum order of $L_1(x)$ and $L_2(x)$.

We arranged the selected test vectors according to Table 2 to prove the CTL reduction in the proposed design. The selected test vectors in each column of Table 2 are not very different. For example, in the second column, $L_1(x)L_2(x)$ is only different in 1 bit with all vectors $\{L_1(x)L_2(x)+1, L_1(x)L_2(x)+X, L_1(x)L_2(x)+X^2, \ldots\}$. Consequently, the resulted fault coverage regarding the test vectors in the same columns does not vary significantly. This point could be interpreted as if any test vector in column *c* of Table 2 occurs, and then we would accept that all test vectors of the column are contributed in the test process, approximately. This proposition leads to a significant reduction in CTL, as described below.

The main modification is related to the *h(i)* calculation. Suppose that the number of selected test vectors in HW-aware design equals the number of columns of Table 2. In HW-aware design, the probability of reception of one selected input vector in every clock would be $1/2^n$. In the CTL-aware design, the hit of a test vector is changed to the hit of a column. Moreover, the test completion of the online approach is changed from receiving all selected test vectors in HW-aware design to receiving at least one test vector from every column of Table 2 in the CTL-aware design.

Ultimately, the probability of receiving a test vector from a specific column would be $2^k/2^n$ and, the related CTL is computed according to (9).

Consequently, the CTL of the proposed CTL-aware design is 2^k times smaller than the HW-aware design.

$$CTL = \sum_1^m \frac{1}{h(i)} = \sum_1^m \left(\frac{2^n}{2^k} \times \frac{1}{i}\right) = 2^{n-k} \times \sum_1^m \frac{1}{i} \tag{9}$$

Assume that m(x) is a polynomial with order *a* (= $\log_2 m$). Then, $m(x).L_1(x).L_2(x)$ has order *n-1* (n is the number

Table 2 Arrangement of test vectors polynomials in the division by $L_1(x)\ L_2(x)$

						R(x)
	0	$L_1(x)L_2(x)$	$x\ L_1(x)L_2(x)$	…	$f(x)\ L_1(x)L_2(x)$	0
	1	$L_1(x)L_2(x)+1$	$x\ L_1(x)L_2(x)+1$	…	$f(x)\ L_1(x)L_2(x)+1$	1
	x	$L_1(x)L_2(x)+x$	$x\ L_1(x)L_2(x)+x$	…	$f(x)\ L_1(x)L_2(x)+x$	x
	x + 1	$L_1(x)L_2(x)+x+1$	$x\ L_1(x)L_2(x)+x+1$	…	$f(x)\ L_1(x)L_2(x)+x+1$	x + 1
G(x) of CTL-aware Test set						
	$x^{k-1}+\ldots.+1$	$L_1(x)L_2(x)+x^{k-1}+\ldots.+1$	$x\ L_1(x)L_2(x)+x^{k-1}+\ldots.+1$	…	$f(x)\ L_1(x)L_2(x)+x^{k-1}+\ldots.+1$	$x^{k-1}+\ldots.+1$

of CUT's inputs). So, the sum of $L_1(x)$ and $L_2(x)$ orders would be n-1-a. As mentioned before, the number of rows in Table 2 (=k) is determined by the minimum order of $L_1(x)$ and $L_2(x)$. The most efficient value for the CTL occurs when both $L_1(x)$ and $L_2(x)$ have equal orders (i.e., k=*(n-a-1)/2*). In this case, the CTL would be calculated using (10).

$$CTL = 2^{n-(n-a-1)/2} \times \sum_1^m \frac{1}{i} = CTL = 2^{(n+a+1)/2} \times \sum_1^m \frac{1}{i} \quad (10)$$

On the other hand, in [28], the practical value for CTL is achieved for n' < 40 in (7). By Comparing (7) with (10), in the proposed design, the upper bound of the input size extends from 40 to 80 – a.

$n_{max}' = (n_{max} + a + 1)/2$
$40 = (n_{max} + a + 1)/2$
$80 = n_{max} + a + 1$
$n > 80 - a$

The CTL-aware design can be implemented by more than two LFSRs. Note that utilization of more than two LFSRs imposes more restrictions on the test vectors. For example, the input vectors which generate an equal remainder in the division by $L_1(x)$, $L_2(x)$, and $L_3(x)$ would belong to the test set. Consequently, the most expanded test set (with the lowest CTL) is achieved when applying two LFSRs for selecting part.

Finding the optimum $L_1(x)$ and $L_2(x)$ is accomplished using the algorithm of Fig. 4. The initial $L_1(x)$ and $L_2(x)$ are selected to satisfy the condition of the mentioned case (i.e., k=*(n-a-1)/2*). Then the multiplication of $L_1(x)$ and $L_2(x)$ ($L_1(x)L_2(x)$) acts as L(x) of Fig. 4's algorithm. If the required fault coverage is achieved, then the algorithm terminates. Elsewhere, the orders of $L_1(x)$ and $L_2(x)$ polynomials are increased, and the previous steps are repeated.

Test vector bit-width reduction and construction of golden circuit (GC) are similar to the HW-aware design counterparts. The only difference is that the number of test vectors in the CTL-aware design equals the total number of vectors in Table 2. By increasing the number of selected test vectors in CTL-aware design, the size of GC would be larger than the HW-aware counterpart. Generally, reducing CTL results in incrementing the hardware overhead in CTL-aware design.

4.2 Offline TPG

In the offline test mode, the MUX_1 passes the output of the offline TPG to the CUT input pins. As mentioned in Sect. 3.1, the input vectors of the first row of Table 1 would act as the pre-computed test vectors. So, we design the offline TPG to generate these test vectors in minimum time.

All pre-computed test vectors are in the form of f(x)L(x), wherein f(x) is the set of all polynomials with order less than or equal n-k. The general form of f(x) of order j is represented in (1). Some terms are missing due to their corresponding c_i being 0. To generate such a polynomial, first of all, $x^dL(x)$ terms should be generated, and these terms should be summed up. Regarding Galois Field algebra, the multiplications of L(x) with x^d (d>0) would be realized by *d* times left shift of binary vector, which represents L(x). Furthermore, the summation of two polynomials in the Galois field can be realized by XORing their corresponding binary vectors.

In the proposed offline TPG design, we store the n-k left-shifted versions of L(x) in different n-bit registers (R_1,.., R_{n-k}). The pre-computed test vectors would be constructed by XORing the contents of the R_i registers. For example, the test vectors in the form of $(X^k + X^{k-1})$ L(x) are generated by XORing R_k and R_{k-1} registers. The test vectors $(X^k + X^{k-2})$ L(x), and $(X^k + X^{k-1} + X^{k-2})$L(x) are generated by XORing (R_K and R_{K-2}), and (R_K, R_{K-1}, and R_{K-2}), respectively.

The schematic of offline TPG for a case with 16 test vectors is illustrated in Fig. 6 (this architecture is used for all cases with 9–15 selected test vectors). There are $4 = \lceil \log_2 T \rceil$ shift registers which store L(x), xL(x), x^2L(x), x^3L(x). The required test vectors (L(x), *x*L(x), *(X + 1)* L(x), …, *$(X^3 + X^2 + X + 1)$* L(x)*)* are generated by insertion of proper XOR gates according to Fig. 6. The last stage of offline TPG is a 16 to 1 MUX and a 4-bit up-counter which passes all 16 test vectors in 16 clock cycles.

The bit-wise XOR of two registers of width W is calculated using a parallel and a serial approach. In the parallel approach, W XOR gates act simultaneously and, the result is generated in a single clock cycle. In the serial approach, the contents of two related registers are copied into two left-shift registers. Then, an XOR gate is deployed to calculate the result in W clock cycles. Although its hardware overhead is more significant than the slower serial approach, the parallel approach is fast.

4.3 Reduction Part

The output of the selecting part in online test mode and the output of offline TPG enters the reduction part. Moreover, the output vector of CUT enters another port of this part. The goal of the reduction part is to compress the width of the selected test vector and the CUT's output vector for simplification of GC design. To achieve this goal, we use two LFSRs. Each of them receives the related vector and generates a remainder according to its characteristic polynomial. The only constraint that should be satisfied is that the generated reminders for selected test vectors in the division by $L_{red}(x)$ should be different.

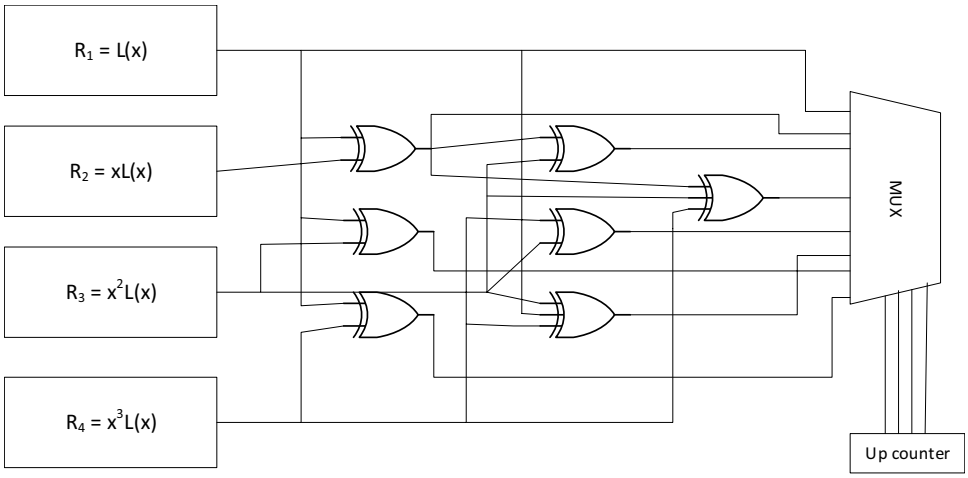

Fig. 6 The test pattern generation (TPG) of the offline part

Notion: if the all-selected test vectors are not divisible by $L_{red}(x)$ (the primary assumption of the notion), then the all-compressed versions of selected test vectors are unequal.

Proof: The division of N_{sel_vec} selected input vectors by $L_{red}(x)$ would result in the following equations, where $Q_i(x)$ and $R_i(x)$ are the quotient and remainder (compressed version of selected input vector) of *i*'th input vector related division.

m-equations

$$L(x) = L_{red}(x)Q_1(x) + R_1(x)$$

$$xL(x) = L_{red}(x)Q_2(x) + R_2(x)$$

$$(x+1)L(x) = L_{red}(x)Q_3(x) + R_3(x) \tag{11}$$

....

$$m(x)L(x) = L_{red}(x)Q_m(x) + R_m(x)$$

We do the proof by contradiction. Assume that two of the remainders are equal:

$$R_i(x) = R_j(x) \tag{12}$$

The related equations for these remainders are:

$$L(x)p(x) = L_{red}(x)Q_i(x) + R_i(x) \tag{13}$$

$$L(x)f(x) = L_{red}(x)Q_j(x) + R_j(x) \tag{14}$$

According to (12) by modular 2 Galois summation concepts:

$$R_i(x) + R_j(x) = 0 \tag{15}$$

Then summation of (13) and (14) is

$$L(x)[p(x) + f(x)] = L_{red}(x)[Q_i(x) + Q_j(x)] \tag{16}$$

$L(x)[p(x) + f(x)]$ is divisible by $L_{red}(x)$

On the other hand, the right-hand side of (17) is one of the selected input vectors. Because the orders of p(x) and f(x) are less than the order of m(x), summation of them in the Galois field will result in a polynomial with order less than or equal to their maximum orders. Due to the presence of all polynomials with orders less than m(x) in the selected input vectors, we can deduce that $L(x)\left[p(x) + f(x)\right]$ would be one of these vectors. Consequently, if two equal remainders exist in m-equations, then at least a test vector will divide $L_{red}(X)$, which contradicts the primary assumption of the notion.

The process of finding $L_{red}(x)$ is as follows. We start with the minimum order polynomial for which the number of different reminders would equal the number of test vectors. The various primary polynomials of the minimum order are tested. If one of these polynomials satisfies the primary constraints, then the $L_{red}(x)$ is found and, the algorithm is terminated. Elsewhere, the order is increased by 1 and, the previous step's process is repeated. The algorithm is terminated when a $L_{red}(x)$ is extracted. In the worst case, there is no reduction and, all test vectors maintain their original bit-width. Even in this case, we would achieve a large hardware overhead reduction. This is because the GC truth table includes N_{test} determined min-terms, and 2^{Ninp}-N_{test} *don't care* min-terms. Such a tremendous number of *don't care* min-terms results in a significant reduction of GC size.

If the CUT has several output pins, then a similar reduction approach can be applied on the output side. The LFSR_red_out in Fig. 1 is related to this case.

4.4 Compressed Golden Circuit Construction

To determine the fail/pass state of the circuit under test (CUT), we must construct a golden circuit (GC) wherein the output value equals the fault-free CUT's output value for

every test vector. If CUT and GC generate equal values for all test vectors, then the system passes the test process. Various approaches have been developed for GC construction. For example, the duplication method consists of an original CUT copy as GC. So, the golden circuit-related overhead is 100%.

If the online test mode is implemented using the HW-aware approach, both online and offline tests use similar test vectors. Consequently, the online and offline golden circuits are merged. An optimum mapping from the compressed version of test vectors and output vectors is derived using conventional synthesizers. The generated circuit acts as the GC_online of Fig. 1.

According to Sect. 3.2, the offline TPG is implemented based on HW-aware approach test vectors. On the other hand, in the CTL-aware approach, the selected test vectors are different from the generated test vectors by offline TPG. Consequently, in the CTL-aware related BIST design, we separate the offline and online GCs. The online GC is synthesized based on the total test vectors of Table 2 and, the offline GC is implemented based on the first row of Table 1. A separate reduction part is constructed for offline test mode, as indicated in Fig. 1.

Based on the mentioned approaches, we can deduce that CTL-aware BIST design imposes a more significant hardware overhead over the HW-aware design. The main reasons are the larger online GC (due to more test vectors of CTL-aware design) and the additional components (offline GC and separate reduction part) regarding offline test mode.

5 The Detailed Design of C6288

The proposed BIST design for the C6288 benchmark circuit is shown in Fig. 7. C6288 consists of 32 inputs, 32 outputs, and 2417 logic gates. The selecting part is designed based on the HW-aware approach. So $LFSR_1$ with L(x) characteristic polynomial is inserted in the selecting part. $LFSR_2$ and $LFSR_3$ in the reduction part compress the 32-bit input and output vectors into 4-bit and 7-bit width vectors, respectively. The related characteristic polynomials are indicated by $L_{red,\ in}(x)$ and $L_{red,\ out}(x)$, respectively.

The golden circuit which maps the compressed version of test vectors to the compact version of the related output vectors is shown in the green-colored part of Fig. 7. Note that the online and offline golden circuits are merged in HW-aware design.

Table 3 reports the hardware overhead of every module in the C6288 BIST design. The hardware cost of every module is calculated using an equivalent gate metric. The comparator of selecting part is a 28-bit input NOR gate but, XOR gates realize the 7-bit comparator.

Based on the result of Table 3, the overhead of the online part is 19.25%. However, the offline part imposes a large hardware overhead. If the parallel approach accomplishes the XORing of the registers, then the total hardware overhead raises to 105.5%. On the other hand, deploying the serial approach would result in a 69% hardware overhead in the cost of more offline test time.

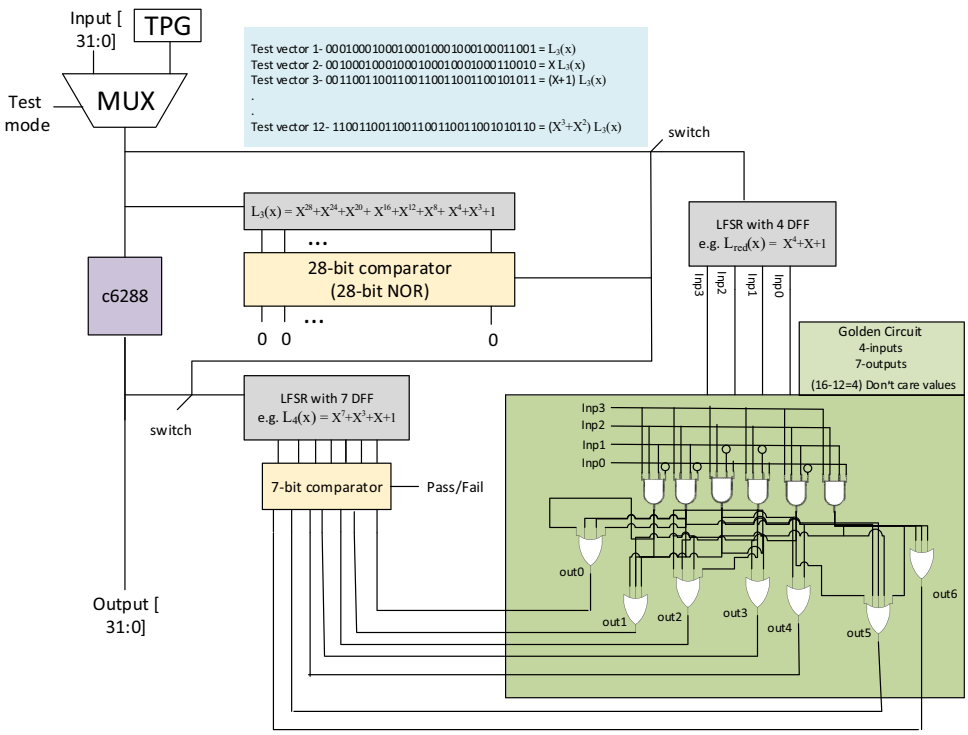

Fig. 7 The proposed design for the C6288 benchmark circuit

Table 3 The hardware overhead for C6288 BIST design.

Module	Number of gates	$HW = \frac{\text{Module gates}}{\text{c6288 gates}} \times 100$
$LFSR_3$	256	10.59
$LFSR_{red}$	40	1.65
$LFSR_4$	68	2.81
7-bit comp	34	1.4
28-bit comp	55	1.15
Golden Circuit	40	1.65
Concurrent	**493**	**19.25**
MUX	128	5.29
TPG 2-bit xor (1)	1052	43.52
TPG 32-bit xor (2)	1920	79.43
Total (1)	**1673**	**69.21**
Total (2)	**2541**	**105.13**

6 Simulation Results and Comparisons

In this section, the proposed method is evaluated by different simulation experiments. Three sets of benchmark circuits (ISCAS85, ISCAS 89, and ITC99) are employed. The hardware overhead and CTL are the major features of every BIST design that must be considered. The hardware overhead is calculated using the gate equivalents metric, where a circuit module (a gate or DFF) is the hardware equivalent of some two-input NAND gates [28]. For example, an N-input OR gate is hardware equivalent of N numbers of two-input NAND gates.

In the following, we compare the proposed BIST design with four online BIST designs: duplication design, MICSET [28], NEMO [25], CE-NEMO [32], and DC-based [13]. Among these state-of-the-art designs, MICSET supports both offline and online tests and, the other ones handle the online test.

6.1 The Overall Results for Benchmark Circuits

To evaluate the performance of the proposed BIST design, the benchmark circuits (ISCAS 85, ISCAS 89, and ITC 99) are divided into three different categories as follows. The small size circuits (SS) contain less than 2000 gates; the large size circuits (LS) have less than 10,000 gates and, the very large size circuits include up to 117,000 gates. The structural information for the benchmark circuits (the numbers of inputs, outputs, and internal modules) are reported in the second, third, and fourth columns of Table 4. The test vectors are derived using the proposed algorithm of Fig. 4.

Table 4 The hardware overhead for various benchmark circuits.

circuit	Inp	Out	Gates (gates+FF)	Test patterns of proposed method		HW-aware HW @ FC (%)		CTL-aware HW @ FC (%)	
				80% FC	90% FC	80%	90%	80%	90%
SS									
c432	36	7	160	42	50	178.26	199.32	210.31	230.98
c499	41	32	202	42	53	98.36	110.65	139.23	147.85
c880	60	26	383	54	62	86.56	95.56	115.69	124.32
c1355	41	32	546	65	73	68.59	82.65	91.13	103.63
c1908	33	25	880	88	98	39.45	47.35	62.35	69.35
c2670	157	64	1193	82	90	95.68	110.65	123.45	132.47
c3540	50	22	1669	120	134	36.97	42.92	65.38	72.86
s298	3	6	133	-	-	-	-	-	-
s344	9	11	175	16	21	52.32	58.2	83.86	93.89
s349	9	11	175	12	21	55.39	63.47	84.25	95.89
s386	7	7	165	-	-	-	-	-	-
s400	3	6	183	-	-	-	-	-	-
s420	19	2	212	44	52	69.75	79.99	90.66	97.02
s444	3	6	202	-	-	-	-	-	-
s510	19	7	217	43	51	73.96	81.69	99.35	110.45
s641	35	24	398	45	50	64.35	70.38	83.65	88.72
s715	35	23	512	45	53	52.69	59.56	72.59	80.25
s1196	14	14	547	104	115	43.93	48.69	69.23	76.32
s1238	14	14	547	106	115	48.56	53.96	68.65	77.32
s1423	17	5	731	54	62	16.45	22.68	45.23	51.73
s1488	8	19	659	-	-	-	-	-	-
s1494	8	19	653	-	-	-	-	-	-
Average						**67.57**	**76.73**	**94.06**	**103.31**
LS									
c5315	178	123	2307	163	175	46.05	53.03	71.63	75.03
c6288	32	32	2417	12	26	19.25	21.69	28.36	33.32
c7552	207	108	3512	178	190	29.18	34.19	43.54	45.2
s5378	35	49	2958	226	252	12.03	15.52	30.36	32.9
s9234	36	39	5825	426	458	7.26	8.47	15.36	19.26
s13207	62	152	8620	315	350	4.69	5.56	21.36	24.2
s15850	77	150	10306	352	394	4.23	5.12	27.56	29.72
Average						**17.57**	**20.49**	**33.88**	**37.09**
VLS									
s35932	35	320	17793	88	97	3.2	3.8	18.69	21.89
s38417	28	106	23815	973	1011	6.52	7	21.3	25.6
s38584	38	304	20705	713	768	6.96	7.74	22.56	26.52
b17	37	97	33741	2286	2352	5.23	5.9	17.56	19.36
b18	37	23	117941	6106	6408	2.23	2.51	12.56	14.23
b19	24	30	-	-	-	-	-	-	-
b20	32	22	20716	1245	1320	6.86	7.23	21.39	23.56
b21	32	22	21061	1263	1380	6.75	7.15	20.96	24.68
b22	32	22	30686	1786	1850	5.12	5.55	15.23	16.35
Average						**5.35**	**5.86**	**18.78**	**21.52**

The test set sizes are reported in columns 5 (for 80% fault coverage) and 6 (for 90% fault coverage), respectively. Obviously, the size of the test set increases with increasing the fault coverage.

In the following columns, the hardware overheads of the online part in the HW-aware and the CTL-aware approaches are reported for 80% and 90% fault coverage values. Some points would be deduced as follows. First, the area overhead of the HW-aware approach is less than the CTL-aware approach in all cases. Because of the larger size of the test set of the CTL-aware approach, the sizes of selecting part and online golden circuit become larger. Second, the hardware overhead increases by a slight increment of fault coverage due to the increment of the test set when the fault coverage increases. Third, our proposed method generally gains a better performance by increasing the size of CUT. This is because of the reduction of the LFSRs impact on hardware overhead for larger circuits. For example, suppose that 3 LFSRs of size N are required for CUT_1, and CUT_2 with gate equivalent sizes 1000 and 10,000, respectively. Then the LFSR-related hardware overhead for CUT_1 and CUT_2 would be 18 N/1000 and

18 N/10000, respectively. The hardware overhead of LFSRs is only 10% for the large CUT_2. Fourth, the hardware overhead is significantly impacted by the numbers of primary inputs and test vectors. For example, C6288 has 32 primary inputs and 12 test vectors, which are significantly smaller than 178 primary inputs and 163 tets vectors of C5315 (the number of test vectors is related to the HW-aware approach). These differences result in a significant reduction of hardware overhead of C6288 compared to C5288 despite that the size of the latter one is slightly smaller than the former circuit. As a worth noting point, the proposed BIST design is based on compaction of the bit-width of the input and output vectors. So, the circuits with the number of inputs less than 8 bits are not included in Table 4. Because such circuits can be tested exhaustively, the test set contains all possible input vectors. As the last point, we conclude that the online part of our proposed BIST design is most beneficial for the moderate and large size circuits and circuits with high primary input size. Figure 8 illusrates the decreasing trends of hardware overhead by increasing the circuit size.

The offline part of the proposed BIST design includes the TPG, MUX_1, and the offline golden circuit. The hardware overhead values are reported in Table 5. In the second column, the numbers of required registers to store the shifter versions of L(x) are reported that do not exceed 11 for very large circuits. The third column indicates the number of required XOR gates for every benchmark circuit.

The fourth column of Table 5 reports the required clock cycles and, the fifth column indicates the hardware overhead of the parallel approach, respectively. The hardware overhead for large circuits is generally below 90% except for C5315 and C7352 circuits as well, as, for very large size circuits, the maximum hardware overhead is 51%. However, the hardware overhead in the case of small circuits is 1000% on average which is due to a large equivalents gate metric of registers and XOR gates compared to the size of CUTs.

The hardware overhead of the serial approach reduces significantly for all cases. The hardware overheads of LS circuits (unless for C5315 and C7352 circuits) are less than 46%. For the VLS circuits, the maximum value reduces to 21%. The same trend persuades SS circuits wherein the average hardware overhead declines to 322.6%. On the other hand, assuming the clock frequency equals 100 MHz, the maximum offline test time for parallel and serial approaches would be 0.0000641 and 0.002371 s, respectively.

6.2 Comparision

Our proposed design is compared with previous methods in three cases. In the first case, the hardware overhead of the online part of the proposed design is compared with NEMO, modified NEMO, and DC designs. All of these designs support only online tests. The comparison results are reported in Tables 6 and 7, wherein the numerical results of previous studies are taken from the published

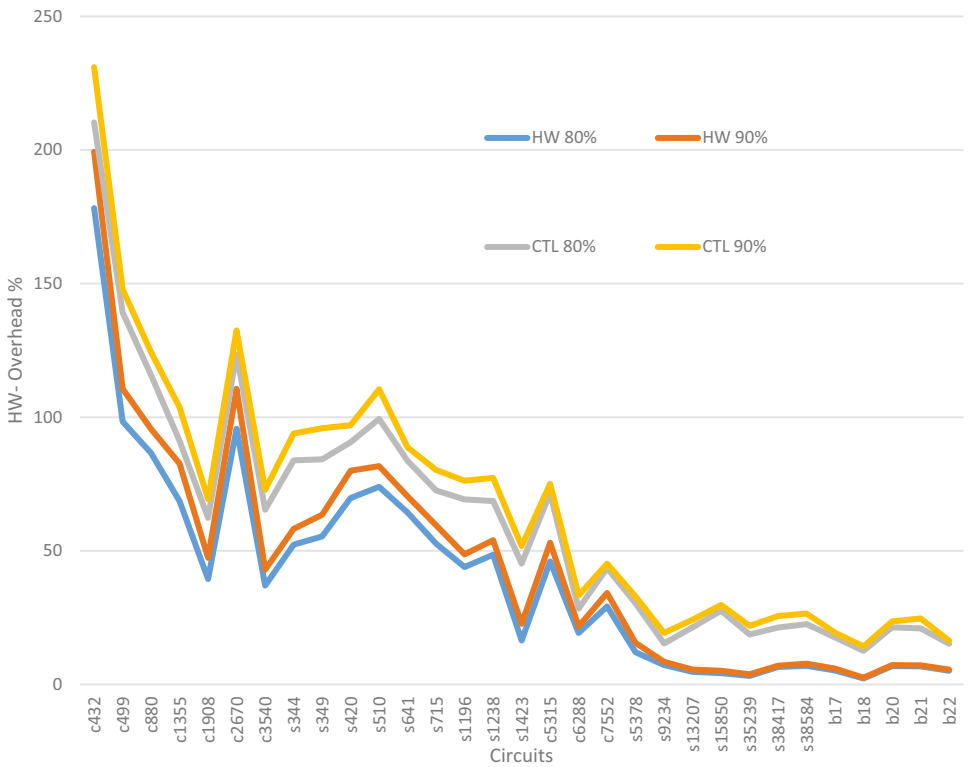

Fig. 8 The HW-overhead trend in the proposed design

Table 5 The hardware overhead for the offline part.

Circuit	required SR = GC's inputs	Required XORs	Required Clocks n-bit XOR	HW (%) n-bit XOR	Required Clocks 2-bit XOR	HW(%) 2-bit XOR
SS						
c432	6	25	50	2087.6	1800	775.3
c499	6	25	53	1084.2	2173	351.6
c880	6	25	62	1973.3	3720	662.2
c1355	7	41	78	1610	3198	439.2
c1908	7	41	110	825	3630	210
c2670	7	41	94	2686	21902	730
c3540	8	63	90	887	4500	179
s298	-	-	-	-	-	-
s344	5	14	21	308	189	136
s349	5	14	21	307	189	135
s386	-	-	-	-	-	-
s400	-	-	-	-	-	-
s420	6	25	52	837.5	988	281.4
s444	-	-	-	-	-	-
s510	6	25	51	844.5	969	285.84
s641	6	25	50	975.6	1750	316.2
s715	6	25	53	950.6	1855	306.5
s1196	7	41	115	457	1610	116
s1238	7	41	115	466.6	1610	112.9
s1423	6	25	42	318.9	714	125.6
s1488	-	-	-	-	-	-
s1494	-	-	-	-	-	-
Average				**1038.44**		**322.6**
LS						
c5315	8	63	175	2238	31150	474
c6288	5	25	26	81.69	862	46.23
c7552	8	63	190	1817	39330	368
s5378	8	63	252	87.77	8820	42.15
s9234	9	93	458	65.31	8702	36.73
s13207	9	93	350	65.96	11550	34.36
s15850	9	93	394	78.26	30338	35.69
Average				**633.42**		**146.73**
VLS						
s35932	7	41	97	30.66	3395	20.36
s38417	10	129	1011	35.33	23808	19.14
s38584	10	129	768	26.66	9216	15.74
b17	12	231	2352	50.99	87024	21.29
b18	13	298	6408	32.12	237096	10.66
b19	-	-	-	-	-	-
b20	11	175	1320	42.23	42240	16.23
b21	11	175	1380	44.36	44160	15.32
b22	11	175	1850	44.82	59200	15.69
Average				**38.39**		**16.8**

papers. So some of the benchmark circuits have been dropped from the tables.

The hardware overheads of the online part of the proposed BIST design and DC-based design are reported in Table 6 (for both cases of fault coverage values (80% and 90%)). The related graphical representation is illustrated in Fig. 9. The DC-based BIST design outperforms HW-aware and CTL-aware approaches for SS circuits. This is due to the tremendous impact of LFSRs on hardware overhead in our design. But for large and very large circuits wherein the LFSRs impact diminishes, the hardware overhead of the HW-aware approach is significantly smaller than DC-based design (approximately 10% on average). Moreover, the CTL-aware hardware overhead approximately equals the DC-based design. The difference between the proposed BIST design and the DC-based one is evenly increased to 35% in the last two rows of Table 6 (s13207 and s15850).

Table 7 and the corresponding illustration in Fig. 10 indicate that both NEMO and CE-NEMO have a significant hardware overhead than the online part of our proposed BIST design. The CE-NEMO reduces the hardware overhead of NEMO by almost 7% on average. However, except for limited cases such as s349 and C6288 circuits, the hardware overhead of CE-NEMO and NEMO are greater than both HW-aware, and CTL-aware approaches. According to Fig. 10, there is a small imrovment in CE-NEMO compared

Table 6 The hardware overhead comparison with DC-based Design.

circuit	HW-aware HW @ FC (%)		CTL-aware HW @ FC (%)		DC-based [9] HW @ FC (%)	
	80%	90%	80%	90%	80%	90%
SS						
c432	178.26	199.32	210.31	230.98	70.3	89.1
c499	98.36	110.65	139.23	147.85	-	-
c880	86.56	95.56	115.69	124.32	55.5	76.6
c1355	68.59	82.65	91.13	103.63	-	-
c1908	39.45	47.35	62.35	69.35	25.7	42.2
c2670	95.68	110.65	123.45	132.47	55.7	-
c3540	36.97	42.92	65.38	72.86	18.3	26.1
s298	-	-	-	-	39.4	55.7
s344	52.32	58.2	83.86	93.89	41.6	48.1
s349	55.39	63.47	84.25	95.89	33.5	45
s386	-	-	-	-	52.3	71.5
s400	-	-	-	-	41.2	53.5
s420	69.75	79.99	90.66	97.02	-	-
s444	-	-	-	-	37	47.6
s510	73.96	81.69	99.35	110.45	-	-
s641	64.35	70.38	83.65	88.72	-	-
s715	52.69	59.56	72.59	80.25	-	-
s1196	43.93	48.69	69.23	76.32	45.2	60.7
s1238	48.56	53.96	68.65	77.32	47.1	62.7
s1423	16.45	22.68	45.23	51.73	47.3	58.5
s1488	-	-	-	-	28.8	38.6
s1494	-	-	-	-	30	38.5
Average	**67.57**	**76.73**	**94.06**	**103.31**	**41.80**	**54.29**
LS						
c5315	46.05	53.03	71.63	75.03	37.8	51.9
c6288	19.25	21.69	28.36	33.32	4.8	6.7
c7552	29.18	34.19	43.54	45.2	32.5	-
s5378	12.03	15.52	30.36	32.9	40.4	55.8
s9234	7.26	8.47	15.36	19.26	29.3	40.2
s13207	4.69	5.56	21.36	24.2	30	39.5
s15850	4.23	5.12	27.56	29.72	28.8	39.4
Average	**17.57**	**20.49**	**33.88**	**37.09**	**29.08**	**38.91**

Table 7 The hardware overhead comparison with NEMO approaches.

circuit	HW-aware 95% FC	CTL-aware 95% FC	NEMO [26]	CE-NEMO [27]
SS				
c432	204.24	239.65	256.9	237.5
c499	116.52	150.47	280.7	258.2
c880	101.69	127.25	249.6	227.6
c1355	85.65	108.69	187.9	157.2
c1908	50.67	75.41	112.8	106.1
c2670	117.93	139.65	436.2	402.4
c3540	44.98	75.96	100	95.5
s298	-	-	85.8	77.5
s344	63.2	95.89	-	-
s349	69.49	97.8	66.3	58.3
s386	-	-	95.2	90.4
s400	-	-	-	-
s420	83.99	99.52	153.2	143.5
s444	-	-	-	-
s510	87.6	114.75	170.3	157.5
s641	79.3	94.82	153.3	138.2
s715	62.56	85.21	-	-
s1196	53.69	78.12	-	-
s1238	56.9	80.32	144.4	136.1
s1423	25.58	53.7	137.5	125.4
s1488	-	-	46.1	42.8
s1494	-	-	-	-
Average	**81.49**	**107.32**	**167.2**	**153.38**
LS				
c5315	55.69	78.56	196.9	185.2
c6288	24.96	35.91	15.1	13.4
c7552	37.55	47.69	224.8	212.7
s5378	17.69	31.24	-	-
s9234	9.98	21.6	-	-
s13207	6.32	26.99	-	-
s15850	6.01	30.72	-	-
Average	**22.6**	**38.95**	**145.6**	**137.1**

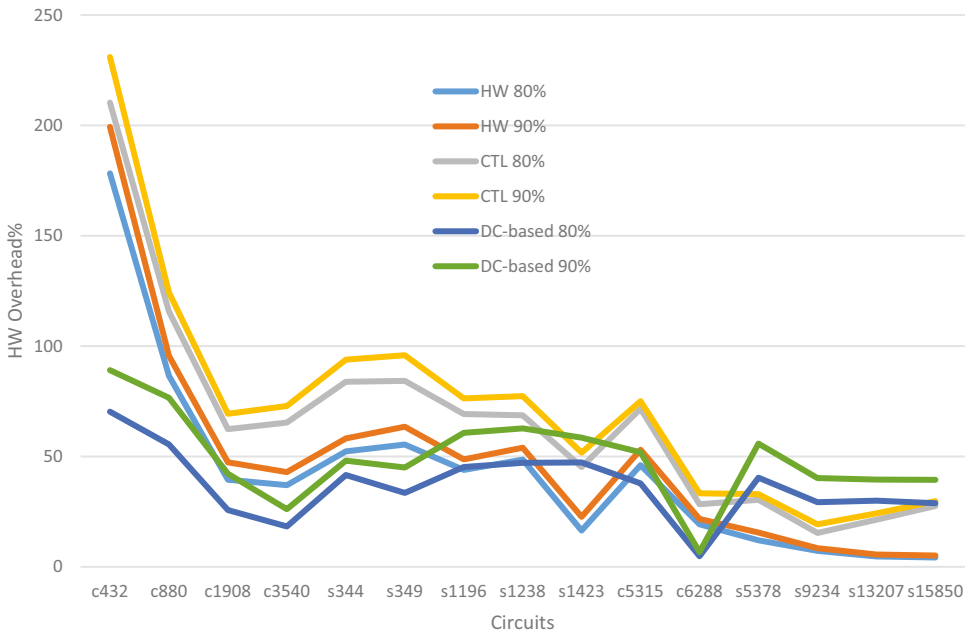

Fig. 9 Comparison of the HW-overhead trend in the proposed design with DC-based approach

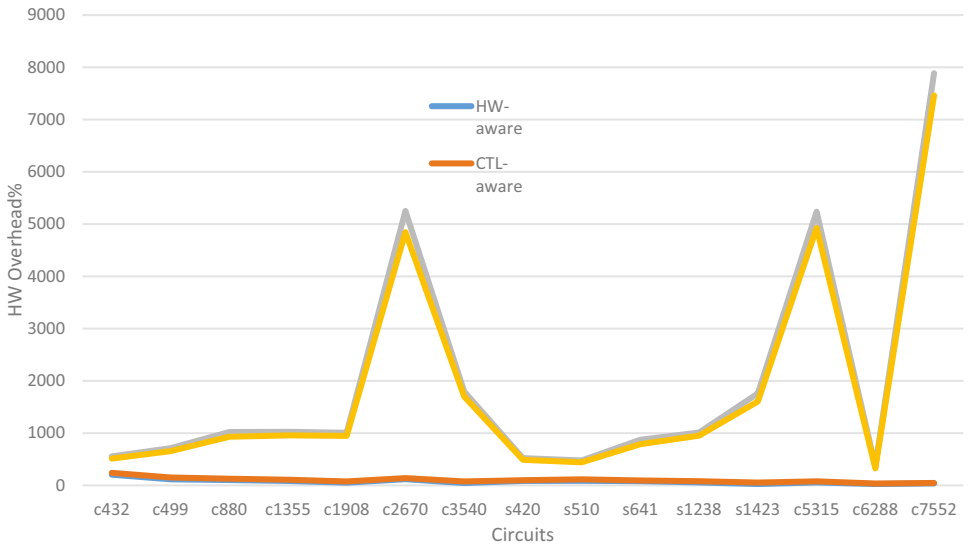

Fig. 10 Comparison of the HW-overhead trend in the proposed design with NEMO and CE-NEMO approaches

to NEMO and the irregular behavior in hardware overhead is apparent for NEMO and CE-NEMO.

As mentioned before, MICSET is the only BIST design that supports both offline and online test modes. The comparison between our proposed BIST design and MICSET is reported in Table 8, wherein the related data of MICSET are taken from [13]. The results demonstrate that the proposed BIST design has a lower hardware overhead. Even for the large circuits in which the hardware overhead of MICSET reduces below 100%, the related hardware overhead of our BIST design would become less than 50%. The graphs for hardware overhead of HW-aware, CTL-aware, MICSET,

Table 8 The hardware overhead comparison with MICSET and the duplication approaches.

circuit	HW-aware 95% FC	CTL-aware 95% FC	MICSET [3]	Duplication
ISCAS 85				
c432	204.24	239.65	1307	125
c499	116.52	150.47	1236	110
c880	101.69	127.25	1283	135
c1355	85.65	108.69	947	104
c1908	50.67	75.41	838	116
c2670	117.93	139.65	-	134
c3540	44.98	75.96	695	117
c5315	55.69	78.56	-	105
c6288	24.96	35.91	79	110
c7552	37.55	47.69	-	122
Average	**83.98**	**107.92**	**631**	**117**

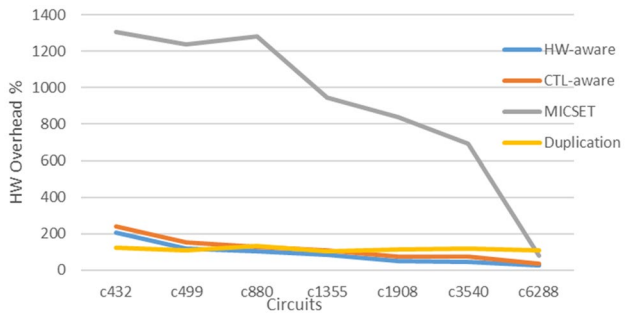

Fig. 11 Comparison of the HW-overhead trend in the proposed design with MICSET and Duplication approaches

and Duplication designs have been plotted in Fig. 11. Our proposed designs outperform the MICSET for small to large circuits and lead to comparable results to the Duplication design (Note that the Duplication method only supports the online test). The MICSET's hardware overhead is significantly reduced for large circuits, but our method achieves an appropriate hardware overhead evenly for small circuits.

Finally, the CTL of HW-aware and CTL-aware approaches are compared with DC-based and CE-NEMO designs in Table 9. Three important points are derived from the results. First, the CTL of CE-NEMO and HW-aware approaches are in the same order. Due to that, these designs are not concerned with CTL and concentrate on hardware's overhead reduction. Second, DC-based design has good performance in reducing both CTL and hardware overhead. Third, our proposed CTL-aware approach outperforms the other three methods regarding CTL. Even though the CTL of the CTL-aware approach is 10^2 times less than the CTL of DC-based design in the worst case. Consequently, the CTL-aware approach reduces the CTL significantly. Furthermore, a similar hardware overhead is achieved in comparison with DC-based design.

Table 9 The CTL comparisons with DC-based and NEMO approaches

circuit	HW-aware CTL	CTL-aware CTL	DC-based [9] CTL	CE-NEMO [27] CTL
c432	3.08 E11	8.3 E06	3.96 E10	3.02 E11
c499	8.79 E12	4.7 E07	4.5 E11	9.94 E12
c880	4.61 E18	3.43 E10	4.93 E11	5.25 E18
c1355	1.15 E13	6.7 E07	4.59 E11	1.11 E13
c1908	4.62 E10	4.1 E06	3.62 E10	4.58 E10
c2670	7.56 E70	5.3 E36	5.72 E11	7.52 E70
c3540	6.27 E15	1.5 E09	1.58 E11	6.26 E15
c5315	2.06 E54	3.96 E28	3.94 E11	2.01 E54
c6288	1.71 E10	1.4 E06	3.11 E10	1.69 E10
c7552	1.28 E63	9.1 E32	6.18 E11	1.25 E63
s298	-	-	9.34 E04	5.23 E 05
s344	5.91 E 07	5.12 E02	3.86 E07	-
s349	5.93 E 07	5.16 E02	3.86 E07	5.86 E 07
s386	-	-	2.79 E04	3.96 E 04
s400	-	-	8.98 E07	-
s420	1.61 E11	2.31 E04	-	1.58 E11
s444	-	-	7.73 E07	-
s510	1.58 E 08	2.36 E04	-	1.56 E 08
s641	8.19 E 16	1.67 E07	-	8.18 E 16
s715	1.37 E 11	2.37 E07	-	-
s1196	2.38 E 10	5.79 E03	5.98 E06	-
s1238	2.40 E 10	5.83 E03	7.72 E07	2.39 E 10
s1423	1.23 E 28	5.11 E04	5.26 E11	1.19 E 28
s1488	-	-	1.17 E05	8.78 E 04
s1494	-	-	1.10 E05	-
s5378	1.41 E11	1.18 E07	6.38 E11	-
s9234	2.76 E11	6.55 E04	6.43 E11	-
s13207	1.84 E19	8.3 E06	6.68 E11	-
s15850	6.05 E23	3.5 E13	6.86 E11	-
s35932	1.43 E11	8.38 E06	-	-
s38417	1.12 E10	2.09 E06	-	-
s38584	2.80 E11	8.19 E03	-	-
b17	5.52 E11	9.49 E06	-	-
b18	5.58 E11	1.34 E08	-	-
b19	-	-	-	-
b20	4.32 E09	1.18 E07	-	-
b21	4.35 E09	1.23 E07	-	-
b22	4.39 E09	1.27 E07	-	-

7 Conclusion

In this paper, a new BIST design has been proposed which supports both offline and online tests-designing a low hardware overhead TPG for the offline part along with an efficient method for compaction of test vectors bit width results in a very low hardware overhead (5% for very large circuit). Furthermore, a CTL-aware test vector selection approach is proposed for which the test time reduces 100 times in average for benchmark circuits. The method is more efficient for large and very large circuits due to a smaller hardware overhead of LFSRs utilized to test vector selection and compression parts.

Funding There is no funding related to this study.

Data Availability The datasets generated and/or analyzed during the present study are available from the corresponding author on reasonable request.

Declarations

Conflict of Interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Abramovici M, Breuer M, Friedman A (1990) Digital Systems Testing and Testable Design. Computer Science Press
2. Acevedo O, Kagaris D (2015) On the computation of LFSR characteristic polynomials for built-in deterministic test pattern generation. IEEE Trans Comput 65(2):664–669

3. Askarzadeh M, Haghparast M, Jabbehdari S (2021) "Power consumption reduction in built-in self-test circuits." J Ambient Intell Humaniz Comput 1–14
4. Biswas S, Das SR, Petriu EM (2006) Space compactor design in VLSI circuits based on graph theoretic concepts. IEEE Trans Instrum Meas 55(4):1106–1118
5. Divyapreethi B, Karthik T (2015) "Input Vector Monitoring Concurrent BIST Architecture using Modified SRAM Cells", ARPN. J Eng Appl Sci 10(9):4042–4046
6. Efanov DV, Sapozhnikov VV, Sapozhnikov VV (2017) "Conditions for detecting a logical element fault in a combination device under concurrent checking based on Berger's code." Autom Remote Control 78(5):891–901
7. Emara AS, Romanov D, Roberts GW, Aouini S, Ziabakhsh S, Parvizi M, Ben-Hamida N (2021) "An Area-Efficient High-Resolution Segmented Σ Δ-DAC for Built-In Self-Test Applications." IEEE Trans Very Large Scale Integr VLSI Syst 29(11):1861-1874
8. Floridia A, Mongano G, Piumatti D, Sanchez E (2019) "Hybrid online self-test architecture for computational units on embedded processor cores." In 2019 IEEE 22nd International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS) 1–6. IEEE.
9. Jahanirad H (2019) Efficient reliability evaluation of combinational and sequential logic circuits. J Comput Electron 18(1):343–355
10. Jahanirad H, Karam H (2017) BIST-based Testing and Diagnosis of LUTs in SRAM-based FPGAs. Emerging Science Journal 1(4):216–225
11. Jahanirad H, Karam H (2018) "BIST-Based Online Test Approach for SRAM-Based FPGAs." In Proc. Iranian Conf. Electrical Engineering (ICEE), pp. 178–183
12. Jurj SL, Rotar R, Opritoiu F, Vladutiu M (2020) "Online Built-In Self-Test Architecture for Automated Testing of a Solar Tracking Equipment." In Proc. IEEE International Conference on Environment and Electrical Engineering and IEEE Industrial and Commercial Power Systems Europe (EEEIC/I&CPS Europe), pp. 1–7
13. Kochte MA, Zoellin CG, Wunderlich H-J (2010) Efficient Concurrent Self-Test with Partially Specified Patterns. Journal of Electric Testing 26(5):581–594
14. Martínez LH, Khursheed SS, Reddy SM (2020) "LFSR generation for high test coverage and low hardware overhead." IET Computers & Digital Techniques 14(1):27–36
15. Murugan SV, Sathiyabhama B (2021) "Bit-swapping linear feedback shift register (LFSR) for power reduction using pre-charged XOR with multiplexer technique is built-in self-test." J Ambient Intell Humaniz Comput 12(6): 6367–6373
16. Nikitha SA, Paulin S, Venkateshwaran SP (2015) "A concurrent BIST architecture for online input vector monitoring." In Proc. International Conference on Science, Technology, and Management, pp. 1411–1488
17. Pavlidis A, Louërat, MM, Faehn E, Kumar A, Stratigopoulos HG (2020) "Symmetry-based A/MS BIST (SymBIST): Demonstration on a SAR ADC IP" In Proc. Design, Automation and Test in Europe Conference and Exhibition (DATE), pp. 282–285
18. Pavlidis A, Louërat MM, Faehn E, Kumar A, Stratigopoulos HG (2021) "SymBIST: Symmetry-Based Analog and Mixed-Signal Built-In Self-Test for Functional Safety." IEEE Trans Circuits Syst I Regul Pap 68(6):2580–2593
19. Roth J (1966) Paul, "Diagnosis of automata failures: A calculus and a method." IBM J Res Dev 10(4):278–291
20. Saluja KK, Sharma R, Kime CR (1987) "Concurrent comparative testing using BIST resources." In Proc. International Conference on Comput Aided Des, pp. 336–339
21. Saluja KK, Sharma R, Kime CR (1987) Concurrent comparative built-in testing of digital circuits. University of Wisconsin, Engineering Experiment Station
22. Saluja KK, Sharma R, Kime CR (1988) "A concurrent testing technique for digital circuits", IEEE Trans. Comput Aided Design Integr Circuits Syst 7(12):1250–1260
23. Sharma R, Saluja KK (1993) Theory, analysis, and implementation of an online BIST technique. VLSI Design 1(1):9–22
24. Shivakumar V, Senthilpari C, Yusoff Z (2021) A Low-Power and Area-Efficient Design of a Weighted Pseudorandom Test-Pattern Generator for a Test-Per-Scan Built-in Self-Test Architecture. IEEE Access 9:29366–29379
25. Voyiatzis I (2012) "Input Vector Monitoring Online Concurrent BIST based on multi-level decoding logic." In Proc. Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1251–1256
26. Voyiatzis I, Efstathiou C (2013) Input vector monitoring concurrent BIST architecture using SRAM cells. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 22(7):1625-1629
27. Voyiatzis I, Halatsis C (2005) "A Low-Cost Concurrent BIST Scheme for Increased Dependability," IEEE Trans Dependable Secure Comput 2(2):150-156
28. Voyiatzis I, Paschalis A, Gizopoulos D, Halatsis C, Makri FS, Hatzimihail M (2008) An input vector monitoring concurrent BIST architecture based on a pre-computed test set. IEEE Trans Comput 57(8):1012–1022
29. Voyiatzis I, Paschalis A, Gizopoulos D, Kranitis N, Halatsis C (2005) A Concurrent Built-In Self Test Architecture Based on a Self-Testing RAM. IEEE Trans Reliability 54(1):69–78
30. Wang R, Chakrabarty K, Bhawmik S (2015) Built-in self-test and test scheduling for interposer-based 2.5 D IC. ACM Transactions on Design Automation of Electronic Systems (TODAES) 20(4):1–24
31. Wang C-H, Hsieh T-Y (2017) On the probability of detection lossless concurrent error detection based on implications. IEEE Trans Comput Aided Des Integr Circuits Syst 37(5):1090–1103
32. Wu TB, Liu HZ, Liu P.X, Guo DS, Sun HM (2013) A cost-efficient input vector monitoring concurrent online BIST scheme based on multi-level decoding logic. J Electron Test 29(4):585–600

Ahmad Menbari received his BS degree in electrical engineering from the Department of Electrical Engineering, University of Kurdistan, Sanandaj, Iran, in 2019. His main research interests are fault-tolerant systems, digital circuit testing, and reliability analysis of logic circuits.

Hadi Jahanirad received his BS degree in electrical engineering from the Department of Electrical Engineering, Khaje Nasir Toosi University, Tehran, Iran, in 2006, and his MS and a Ph.D. degrees from Iran University of Science and Technology, Tehran, Iran in 2008 and 2012, respectively. Since 2013, he has been with the Department of Electrical Engineering, University of Kurdistan, Sanandaj, Iran, where he is now an assistant professor. His main research interests are digital system design, VLSI design, reliability analysis of logic circuits, digital circuit testing, approximate computing, and evolutionary computing.