Tri-State Coding Using Reconfiguration of Twisted Ring Counter for Test Data Compression

Sungyoul Seo, Yong Lee, and Sungho Kang, *Member, IEEE*

***Abstract*—As technology processes scale up and design complexities grow, system-on-chip integration continues to rise rapidly. According to these trends, increasing test data volume is one of the biggest challenges in the testing industry. In this paper, we present a new test data compression method based on reusing a stored set with tri-state coding (TSC). For improving the compression efficiency, a twisted ring counter is used to reconfigure twist function. It is useful to reuse previously used data for making next data by using the function of feedback of the ring counter. Moreover, the TSC is used to increase the range information transmission without additional input ports. Experimental results show that this compression method improves a compression ratio and a test time on both International Symposium on Circuits and Systems'89 and large International Test Conference'99 benchmark circuits in most cases compared to the results of the previous work without a heavy burden on the hardware.**

***Index Terms*—Automatic test equipment (ATE), logic testing, system-on-chip (SoC), test data compression, tri-state coding (TSC), tri-state detection, tri-state input, twisted ring counter (TRC).**

I. Introduction

DUE TO innovations in the manufacturing technology of system-on-chips (SoCs), more intellectual property cores and modules can be integrated into a single chip. In large designs such as SoCs, attaining a high-test quality requires more test patterns targeting delay faults and other fault models beyond stuck-at faults [1]. As the test data volume increases, memory modification of automatic test equipment (ATE) is required as well as additional test application time (TAT). As a result, the cost to sufficiently test SoCs increases.

There are two solutions to overcome the above problems: 1) built-in self-test (BIST) and 2) test data compression. The former solution needs no external tester, but it is less appropriate for logic testing than memory testing. It leads to inadequate test coverage because of its random-resistant fault and bus contention during the test application [2]. For this reason, a variety of test data compression methods have been developed and used by most SoC designers.

To achieve this, the general test data compression schemes are required to have additional decompression hardware in the SoC and they are necessary to operate this method by the external tester. In spite of these disadvantages, the test data compression is the most preferred test method in the industry because it is compatible with the conventional design rules and is even suitable to the scan testing [1]. In addition, it can reduce the amount of the test data required for an external tester such as an ATE, improve the TAT and maintain high-fault coverage [3]. The compression schemes are stored in the ATE and the test data are transmitted onto the circuit under test (CUT) through test ports, i.e., test data input (TDI) ports.

Test data compression schemes can be classified into three categories: 1) code-based; 2) linear-decompression-based; and 3) broadcast-scan-based [1]. The linear-decompression-based scheme decompresses test data with linear-feedback shift registers (LFSRs), ring generators or exclusive OR (XOR) networks [4]. This technique can generate a test cube (with many don't care bits) using an LFSR with a compact seed or by using a simple XOR network [5]. It takes advantage of the fact that typical scan test patterns have very few specified bits, hence, most test patterns are not specified, i.e., don't care [6]. The broadcast-scan-based scheme broadcasts few control bits and generates a large number of bits to scan chains [7]. A TDI that is scanned in through a scan input of a tester is shared among multiple scan cells [8]. Moreover, this scheme is much simpler than the linear-decompression-based scheme.

Although the linear-decompression-based and the broadcast-scan-based schemes can often acquire better compression efficiency and are available on commercial tools [1], they need some circuit structural information for automatic test pattern generation (ATPG) or fault simulation [9]. Additionally, the ATPG for these schemes generate more test patterns than the code-based scheme because their decompressor cannot make the exact intended patterns for detecting faults due to dependency of the decompressor structure. Moreover, these problems produce many switching activities and experience large power consumption issues, which degrade the test reliability.

The code-based scheme compresses test data using a number of codewords, which are made up of binary bit streams based on some specific properties of the underlying test data [9]. This scheme has an on-chip decoder, which decodes the compressed test data from the ATE and delivers the

Manuscript received June 2, 2014; revised September 5, 2014 and December 16, 2014; accepted February 23, 2015. Date of publication March 16, 2015; date of current version January 19, 2016. This work was supported by the National Research Foundation of Korea (NRF) grand funded by the Korea government, Ministry of Science, ICT and Future Planning (No. 2012R1A2A1A03006255). This paper was recommended by Associate Editor A. E. Gattiker.

The authors are with the Department of Electrical and Electronics Engineering, Yonsei University, Seoul 120-749, Korea (e-mail: shkang@yonsei.ac.kr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2015.2413416

decompressed test data to the scan cells [10]. This on-chip decoder can be easily designed because it has a simple structure such as a small finite state machine and some controllers. Hence, this scheme makes them more applicable for designs.

The remainder of this paper is organized as follows. Section II describes the preliminaries of related works in the code-based test data compression, the tri-state input test data, the reusable characteristic of a shift register, and the test data analysis, and we also present the motivation. In Section III, the proposed compression method, tri-state coding (TSC), is introduced. Section IV describes the proposed decompression architecture that is composed of a tri-state detector, a decompressor, a reconfigured twisted ring counter (R-TRC), and a scan chain. The experimental results are shown in Section V, and we conclude this paper in Section VI.

II. Preliminaries and Motivation

A. Code-Based Test Data Compression

The code-based scheme has been researched over the years to reduce the required test data and decrease the test time with smaller hardware in comparison to other schemes. Both the amount of test data and TAT are major issues for SoCs from an economic testing point of view [15]. At this point, the code-based test data compression is one of the best solutions for the problems. This scheme encodes test cubes which consist of test data with unspecified bits.

Huffman coding is known to be the most effective statistical-based coding method because it is proven to provide the shortest average codeword length among all uniquely decodable variable length codes [13]. Some Huffman coding-based compression methods are published in [14]–[18]. Although there are a high-compression ratio, making a Huffman tree requires large internal hardware. The run-length-based compression method encodes runs of patterns as values and counts [9]. Examples of these method include Golomb coding [19], cyclical scan register (CSR)-based run-length code [20], frequency-directed run-length (FDR) code [3], extended FDR code [22], and dual-run-length code [2]. For improving compression efficiency, data-independent pattern run-length (DIPRL) [23] has been proposed. It takes advantage of equality and complementarity of patterns, but it requires a large hardware area overhead. Similar to DIPRL, 2^n-PRL [9] compression has been published recently; it encodes $2^{|n|}$ runs of compatible or inversely compatible patterns. It has the advantage of simple and easy decompression logic, but it still does not have sufficient compression efficiency. Another method is the dictionary-based code; it is researched in [24]–[26]. This method is useful for the embedded systems domain because it provides good compression efficiency as well as a fast decompression mechanism [10]. Recently, a more optimized version was proposed in [7] and a mixed bit-mask method was proposed in [10]. A new method based on the reuse of parts of dictionary entries [11] considerably improves the test compression efficiency, but it still requires quite a large amount of internal memory for storing some dictionary patterns. From a hardware area overhead point of view, the fact that the dictionary overhead is a large burden cannot be overlooked.

In this paper, we use feedback and twist characteristic of the R-TRC for reusing previously used test data. Moreover, the inserted test data used in the proposed method are tri-state levels. In Sections II-B and II-C, we describe more details of these characteristics. This method overcomes the high-hardware area overhead of the dictionary-based and Huffman methods and the low-compression ratio of the run-length-based method.

B. Input Test Data Consisting of Tri-State Values

As previously mentioned, the input test data which are stored in the ATE memory is generally composed of binary value ("0" and "1"). Although most ATE can support the Hi-Z value transmission from its connected ports, the formal test data compression methods have never attempted to use the Hi-Z values in the design-for-testability (DFT) field. However, we initially use the Hi-Z value for improving the compression efficiency that is beneficial to the environment-limited channel bandwidth. Hence, entropy can be increased for this situation.

In order to communicate with channels using three levels of information, some tri-state detection circuits have been researched in [12] and [27]–[29]. To apply the DFT flow, we select the circuit from [29]; it can be easily combined with the existing decompressor. This circuit requires only six transistors; hence it has a low-area overhead and complexity. The operation of this circuit is very simple. The truth table [29] shows that the output values are determined according to input values. Because most internal digital logic has difficulty handling the Hi-Z value, it is necessary to insert a binary conversion module between the input ports and decompressor. In [29], this circuit converts a bit with a tri-state value to two bits of data with a binary value.

Adding this circuit for the DFT application is very useful, especially in cases with a dependence on the ATE such as the test data compression method. For this, there is no additional cost for modifying the external tester because most existing ATE generally supports Hi-Z output. Besides, it overcomes the restriction in the number of TDI ports, published to the standard of IEEE 1149.1 [30]. As a result, this circuit enables the use of tri-state level input data in order to insert the compressed test data and control the decompressor.

C. Characteristic of the Twisted Ring Counter for Reuse

In the code-based test data compression, there are a variety of methods for reusing stored data. An example is the CSR [17], [20] which added a XOR gate and registers before the CUT. Although the CSR is useful for reducing the amount of test data by making a longer run length through using difference test data, it is not practical to use this module in the data compression architecture [33]. This is because the hardware area overhead grows by the length of the scan chain [34]. An additional application is widely used by LFSRs [35], [36] for the linear and broadcast-based test compression schemes. It consists of a shift register and extra XOR gates; therefore, it is used for the pseudo-random sequence generator. However, this approach is limited for the reuse

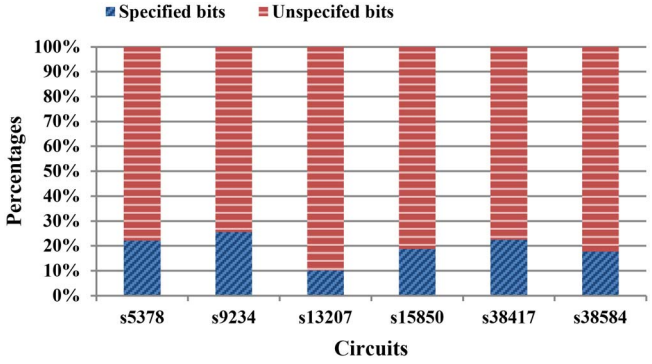

Fig. 1. Test data compositions of ISCAS'89 circuits.

concept because it is dependent on the longer length shift registers for a high-feedback probability.

To overcome the problems discussed above, the R-TRC is used to improve reusable data. The R-TRC has been used to generate test data for test data compression [37] and BIST [38] by reconfiguring the property of it. Flip-flops of ring counter in the TRC can be used to store the test data. However, if the compression ratio is proportional to the length of the R-TRC, inserting a chip for the test is not useful because the hardware area overhead significantly increases. Therefore, considering the compression ratio and the hardware area overhead, this length needs to be calculated appropriately. In Section V, we present this relationship and propose an acceptable length.

D. Test Data Analysis

Since test patterns extracted from the ATPG have many unspecified bits, an important advantage exists that makes the test patterns flexible for improving the compression efficiency. According to our test data analysis of the International Symposium on Circuits and Systems (ISCAS)'89 circuits [31], it has been observed that the test data contain a number of unspecified bits, as shown in Fig. 1. A deterministic ATPG algorithm of TetraMAX is used in a test generation tool of Synopsys. In most of the ISCAS'89 circuits, over 74% of the bits are unspecified. Even the care bit density of most test sets is 1%–5% [1], [32] in the industrial circuits. Moreover, most unspecified bits are directed specifically at the deterministic patterns, which take over 80%–90% of the latter test patterns. Therefore, most test sets (about 80%–90%) can easily produce similar content.

E. Motivation

As mentioned above, although many test data compression methods have been developed, the test data compression method is restricted to a few approaches: 1) run-length; 2) Huffman; and 3) dictionary. These methods cannot completely consider the three main issues for test data compression: 1) the compression ratio; 2) the hardware area overhead; and 3) the TAT. Hence, more than one of these issues tends to be ignored.

The main purpose of a new method is to understand and overcome these problems in contrast with the previous works. For this reason, we improve the reusing of the test data to use the characteristics of the ring counter in the R-TRC and apply the tri-state input data using the tri-state detector. These methods are possible to acknowledge the three main issues at the same time. While the compression ratio and the TAT are better, the increase in the hardware area overhead is very small. The achieved experimental results are presented in Section V.

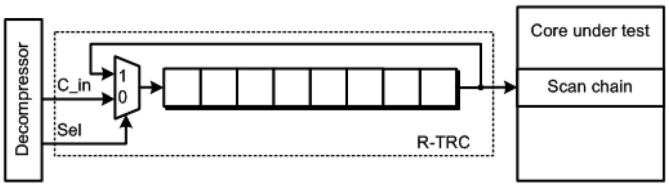

Fig. 2. Simple test architecture using the 10-bit R-TRC.

Procedure of TSC Compression Algorithm

1: Extract test data from the ATPG.
2: Split test data into the length of the R-TRC.
3: Match the split test data with the unspecified bits.
4: Compress each split test data into TSC using the R-TRC.

Fig. 3. Procedure for the proposed test data compression based on TSC.

III. Compression Method

We propose a new test data compression method using the TSC that maximizes reusing frequencies of the stored split data. The R-TRC is composed of a ring counter and a multiplexer (MUX) as shown in Fig. 2. The MUX enables both feedback and twist mode as needed. In addition, the twisted data can be selected as C_in data. Hence, this module can save and change the internal data according to the MUX selection. Section IV discusses the details. The compression procedure following the TSC is conceptually illustrated in Fig. 3. Note that, the extracted test data from the ATPG are split into the length of the R-TRC. Then, each split data is matched with the unspecified bits. Finally, full filled split data are compressed into tri-state code using the R-TRC. While the length of the compressed test data is variable, the decompressed test data is generated at a fixed length. Hence, it is based on the variable-to-fixed compression method. To explain in more detail, we sequentially demonstrate this compression flow and illustrate its examples.

A. Test Data Split

The original test data set, T_D, for a circuit with n test patterns, which are generated by the ATPG, can be represented by an n-tuple set, $T_D = \{T_D^1,\ T_D^2,\ \ldots,\ T_D^n\}$, where the length of ith test data, T_D^i, which consists of 0s, 1s, and Xs, match the length of the scan chain, l_{SC}. After acquiring the above test data, T_D, the next strategy is for each T_D^i to be split into the length of the R-TRC, l_{SR}. Hence, this makes the $m \times n$ split test data sets, $T_D^i = \{T_d^{(i-1)m+1},\ T_d^{(i-1)m+2},\ \ldots,\ T_d^{im}\}$, where each T_d^i is a subset of T_D^i and the length of these subsets is l_{SR}. Hence, test data T_D is composed of the subsets of T_d^i with slices of $m \times n$, as shown in Fig. 4.

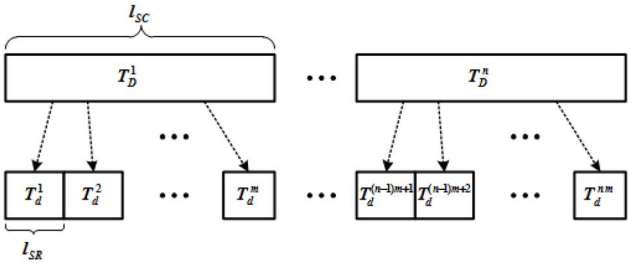

Fig. 4. Test data split into the length of the R-TRC.

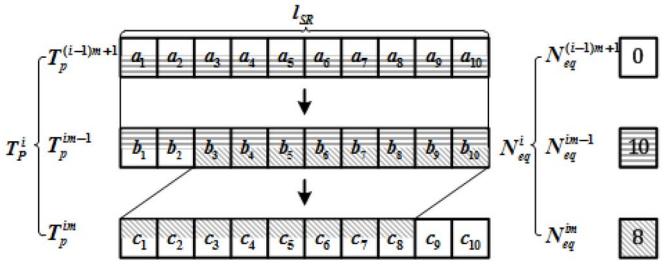

Fig. 5. Forward X-filling for making more matching data.

TABLE I
SIMPLE EXAMPLE FOR SPLIT TEST DATA

Set	Original test data		Bits
T_D^1	0XXX010XXX X01X0XXXXX X11XXX0XXX		30
T_D^2	10XXX0XXXX XXXX00X101 11XXXXXXXX		30
Set	Subsets	Split test data	Bits
T_D^1	T_d^1	0XXX010XXX	10
	T_d^2	X01X0XXXXX	10
	T_d^3	X11XXX0XXX	10
T_D^2	T_d^4	10XXX0XXXX	10
	T_d^5	XXXX00X101	10
	T_d^6	00XX1XXXXX	10

Let us consider the following case where $l_{\mathrm{SC}} = 30$ and $l_{\mathrm{SR}} = 10$; the example is illustrated in Table I. Here, the value of l_{SR} must be determined by a certain standard because this length is associated to the hardware area overhead and compression ratio. The amount of compressed test data can be predicted for determining a suitable l_{SR} by the following:

$$C_B(T_D) \cong |T_D| \times \left[1 - \mathcal{P}_m\left(T_d^i, T_d^{i+1}\right)\right] \times R_{\mathrm{pi}} + m \times n \quad (1)$$

where $C_B(T_D)$ is the number of compressed bits of the test data, T_D, $|T_D|$ is the size of the test data, and $\mathcal{P}_m(T_d^i, T_d^{i+1})$ is the probability of match between T_d^i and T_d^{i+1}; it can be written as follows:

$$\mathcal{P}_m\left(T_d^i, T_d^{i+1}\right) = \frac{1}{n \times m - 1} \sum_{i=1}^{n \times m - 1} \left(T_d^i == T_d^{i+1}\right) \quad (2)$$

where $(T_d^i == T_d^{i+1})$ is 1 if T_d^i can match T_d^{i+1} completely; otherwise, it is 0. R_{pi} is the ratio of the partial input insertion data for changing the stored the R-TRC data when $(T_d^i == T_d^{i+1})$ is 0. R_{pi} can be roughly regarded as constant at 0.48, which is a heuristic result by our experiment. The last term, $m \times n$, is the number of the split test data. The Hi-Z signal is always assigned to the end of the split test data set, T_d^i, to announce the end of the insertion in the R-TRC. A more specific reason is illustrated in Section IV.

Finally, the key is that the two terms, $\mathcal{P}_m(T_d^i, T_d^{i+1})$ and m, are dependent on l_{SR}. Therefore, l_{SR} is closely related to the compression ratio and hardware area overhead. The effort to obtain a saturated area for proper l_{SR} is important for acquiring improved results. The accuracy of (1) is specified for the experimental results in Section V.

TABLE II
SIMPLE EXAMPLE FOR FORWARD X-FILLING

Set	Subsets	Partially filled split test data	N_{eq}	Bits
T_P^1	T_p^1	0XXX010XXX	0	10
	T_p^2	**00**1X0**10**XXX	10	10
	T_p^3	**0110100**XXX	9	10
T_P^2	T_p^4	10**100**0XXXX	8	10
	T_p^5	**1010**00X101	10	10
	T_p^6	00**0**X1**01**XXX	7	10

B. Matching the Split Test Data With the Unspecified Bits

After producing the split test data, each split test data needs to be made to a lot more matching data by filling the unspecified bits to 0 or 1 because the R-TRC efficiently improves the compression ratio with them. In order to compress the split test data, the two steps should be performed: 1) a forward X-filling for more matching data and 2) a backward X-filling for using the R-TRC. The unfilled split test data, T_d^i, are converted to forward X-filled split data, T_p^i, where $T_P^i = \{T_p^{(i-1)m+1}, T_p^{(i-1)m+2}, \ldots, T_p^{im}\}$; they are a subset of T_P.

Additionally, the variable N_{eq} is added to store the matching point compared to a previous test data and this point is the number of matching bits and is used in the next step for encoding the data. Let us assume that l_{SC} is 30 and l_{SR} is 10, the forward X-filling method is represented in Fig. 5, where x_i indicates a binary or an unspecified bit. This method is preceded in the forward direction following the arrow. If it is a first test data, X-filling is not applied to this data and N_{eq} is assigned 0. From the second test data to the last test data, searching the matching part between the current split test data and the previous one is conducted iteratively; the unspecified values of the current matching part are assigned to the values of the previous matching part, where the matching part indicates that they can be equal. The forward X-filling algorithm is illustrated in Fig. 6 and an example of this is shown in Table II. In Fig. 6, the fifth line detects whether the selected part is matched or not; the size of the equal part is stored in N_{eq} through the sixth and ninth lines fulfill the X-filling. Lastly, the unequal part is retained as shown in 14th line. This paper is iterated; hence the split test data of Table I are converted to the partially filled split test data of Table II through the proposed algorithm.

Algorithm for forward X-filling split test data

1: Initialize partial X-filling test set, T_P
2: $T_p^1 \leftarrow T_d^1$
3: **for** i=1 to $mn-1$ **do**
4: **for** l=1 to l_{SR} **do**
5: **if** T_p^i bit streams from lth to l_{SR}th are equal to T_d^{i+1} bit streams from 1st to $(l_{SR}-l+1)$th **do**
6: $N_{eq}^{i+1} \leftarrow l_{SR}-l+1$
7: **for j=1 to** $l_{SR}-l+1$ **do**
8: **if** $T_p^{i+1}[j]$ is an unspecified bit **do**
9: $T_p^{i+1}[j] \leftarrow T_p^i[j+l-1]$
10: **else do**
11: $T_p^{i+1}[j] \leftarrow T_s^{i+1}[j]$
12: **end if**
13: **end for**
14: $T_p^{i+1}[l_{SR}-l+2] \sim T_p^{i+1}[l_{SR}] \leftarrow T_d^{i+1}[l_{SR}-l+2] \sim T_d^{i+1}[l_{SR}]$
15: **break for**
16: **end if**
17: **end for**
18: **end for**

Fig. 6. Algorithm for forward X-filling split test data.

TABLE III
SIMPLE EXAMPLE FOR BACKWARD X-FILLING

Set	Subsets	Fully filled split test data	N_{eq}	Bits
T_F^1	T_f^1	0**011**010**000**	0	10
	T_f^2	001**1**010**000**	10	10
	T_f^3	0110100**001**	9	10
T_F^2	T_f^4	1010000**101**	8	10
	T_f^5	1010000101	10	10
	T_f^6	0000101**111**	7	10

In contrast to a previous step, the order of the next X-filling is backward; hence this paper is performed from the last to first data. To be more specific, T_P^i is converted to the backward X-filled split test data, T_F^i, and T_F^i is encoded to the final encoded split test data, T_E^i, where $T_F^i = \{T_f^{(i-1)m+1}, T_f^{(i-1)m+2}, \ldots, T_f^{im}\}$ is a subset of T_F. This method is the procedure for using the R-TRC.

An example is shown in Table III and the following data are the results applied from Table II. First of all, the last data, T_P^6, is applied to the adjacent filling, which is assigned a previous bit if there is an unspecified bit. Next, the content between $T_f^5[l_{\mathrm{SR}} - N_{\mathrm{eq}}^6 + 1]$ and $T_f^5[l_{\mathrm{SR}}]$ is assigned, in order, to the content from $T_f^6[1]$ to $T_f^6[N_{\mathrm{eq}}^6]$. On the other hand, the other contents are still applied to the adjust filling. When this procedure reaches T_f^1, the full X-filling test data, T_F, can be acquired.

C. TSC

The final step is that the results so far, T_F^i, are encoded to TSC, T_E^i, using the R-TRC, where $T_E^i = \{T_e^{(i-1)m+1}, T_e^{(i-1)m+2}, \ldots, T_e^{im}\}$ is a subset of T_E. Here, T_e^i has variable length and it can be found to be $l_{\mathrm{SR}} - N_{\mathrm{eq}}^i + 1$ and this content consists of the R-TRC insertion values from $T_f^i[N_{\mathrm{eq}}^i + 1]$ to $T_f^i[l_{\mathrm{SR}}]$ and a Hi-Z signal.

TABLE IV
SIMPLE EXAMPLE FOR ENCODING TEST DATA

Set	Subsets	Encoded test data		Bits
		R-TRC insertion data	Signal	
T_E^1	T_e^1	0011010000	Z	11
	T_e^2		Z	1
	T_e^3	1	Z	2
T_E^2	T_e^4	01	Z	3
	T_e^5		Z	1
	T_e^6	111	Z	4
Total (bits)		16	6	22

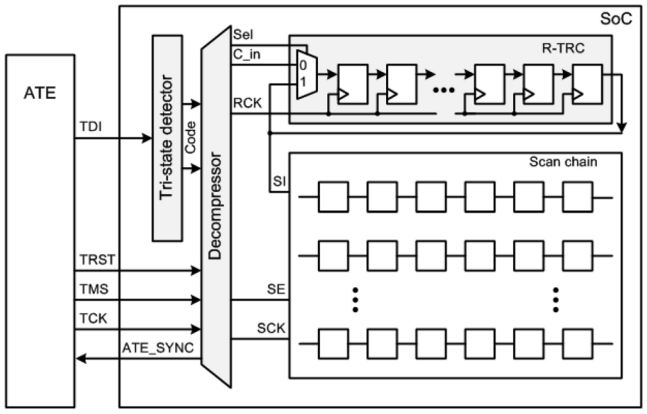

Fig. 7. Conceptual overview of the proposed decompression architecture in a single chain environment.

An example for encoding test data is shown in Table IV. The data shown in Table IV is encoded from the fully filled test data shown in Table III. First of all, the first encoded data, T_e^1, is composed of T_f^1 and a Hi-Z value. However, the next T_e^2 has only a Hi-Z because the next T_f^2 is exactly equal to T_f^1. In case of T_e^3, it is composed of 1 and a Hi-Z value. It means that if 1 is inserted to the R-TRC, it can make T_f^3 data because the R-TRC is stored to T_f^2 data. In Table IV, the third column data is essentially inserted contents to make the test data. This procedure is iterated until T_e^i reaches the last encoded test data. As a result, the final compressed data size is 22-bit, which is composed of binary and Hi-Z values.

IV. DECOMPRESSION ARCHITECTURE

The proposed decompression architecture in the single chain environment is given in Fig. 7. In a conventional test flow, compressed test data (codewords) that consists of 0s and 1s are initially transmitted to an on-chip decoder then the decoder decodes them and delivers the original test data to the scan cells [10]. However, the proposed architecture is required to have a tri-state detector behind the TDI and a R-TRC

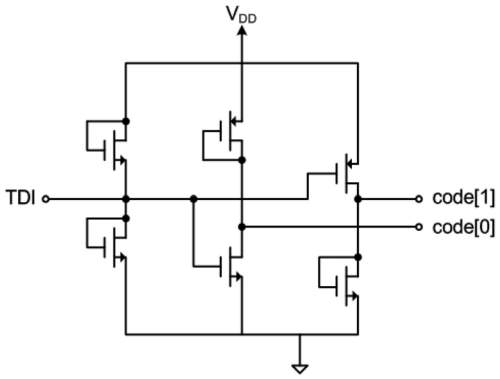

Fig. 8. Tri-state detector proposed by Thomson *et al.* [29].

TABLE V
TRUTH TABLE OF TRI-STATE DETECTOR

Inputs	Outputs	
TDI	code[1]	code[0]
'0'	'1'	'1'
'1'	'0'	'0'
Hi-Z	'1'	'0'

behind the decompressor. To be more specific, each module is continually represented in this section.

A. Tri-State Detector

The added tri-state detector is important for increasing the range information transmission from the ATE without additional ports. Hence, this module enables to improve the entropy from 1- to 1.5-bit. This range expandability can be widely used for applying the general decompression architecture for the same purpose. In Fig. 8, the tri-state detector outputs 2-bit results through code ports; thereby the written TDI that is composed of a tri-state level is converted to 2-bit binary value as shown in Table V. The objective of this conversion is for the Hi-Z value to not be recognized by the decompressor and the registers. This tri-state detector module is required to have only six transistors, hence it is a negligible burden compared to the expected effects.

B. Decompressor

The code-based schemes usually have a complicated design flow due to the decompressor. For this reason, the proposed decompressor has a simple structure and lessens the dependency of the test data. Generally, the internal hardware for DFT should be significantly modified when the test data is changed in the code-based schemes. However, since the proposed method requires a slight modifications for redesign when test data or the target circuit is changed, the design burden and complexity of the design flow of the proposed method is less than compared to other code-based schemes. The proposed decompressor is given in Fig. 9; it is an internal architecture of the decompressor in Fig. 7. This decompressor is comprised of a code converter, a k-bit counter, and a control and generator unit (CGU). The code converter has simple combinational logic in order to identify the code data that are outputs of the tri-state detector. It has two input and output

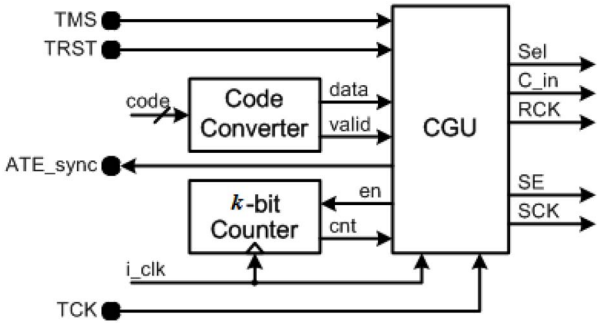

Fig. 9. Decompressor for TSC using the R-TRC.

TABLE VI
TRUTH TABLE OF CODE CONVERTER

Inputs		Outputs	
code[1]	code[0]	data	valid
'1'	'1'	'0'	'1'
'0'	'0'	'1'	'1'
'1'	'0'	'0'	'0'

ports and its truth table is shown in Table VI. The k-bit counter helps to control the CGU to enable the shifting operation from the R-TRC to the scan chain, where k is $\lceil \log_2(l_{SR} + 1) \rceil$. The CGU is responsible for controlling the R-TRC and scan chain, delivering the decompressed test data to the scan chain, adjusting the suitable clocks among the ATE and internal clocks, and synchronizing the clocks between the ATE and internal circuit.

For example, if the valid is 1, this means that the next split test data is required to change the R-TRC data. Therefore, C_in is set to the same value of the data and the selection signal (Sel) of the MUX is set to 0. Furthermore, C_in value is transmitted to the shift register of the R-TRC by operating test clock input, which is set to the port of the R-TRC clock (RCK). On the other hand, the other output ports of the CGU are assigned 0. Another example is that if the valid is 0, the stored the R-TRC data are sent to the scan chain through the scan data in (SI) and feedback operation is carried out simultaneously by changing the Sel from 0 to 1. For a fast sending operation, the RCK and scan clock are set to the internal clock (i_clk). Furthermore, the scan enable (SE) and the counter enable (en) are assigned 1. This sending operation proceeds until the result of the counter (cnt) is l_{SR}. The two operations above are iterated until the scan test is finished.

C. R-TRC

The R-TRC is a module that stored the previous inserted test data so that future requests for that data can be reused as if it is a cache in the computer architecture. This module is similar to a cache memory in computer architecture. The required data from the ATE can be minimized by reusing and duplicating the data that are stored in the R-TRC. For using this reuse stored data, all test data must be conveyed to the scan cells through the R-TRC as mentioned in Section IV-B. Therefore, once the R-TRC is completely filled with valid test data, these data are fed back to this module and transmitted to the scan cells simultaneously by operating an internal clock.

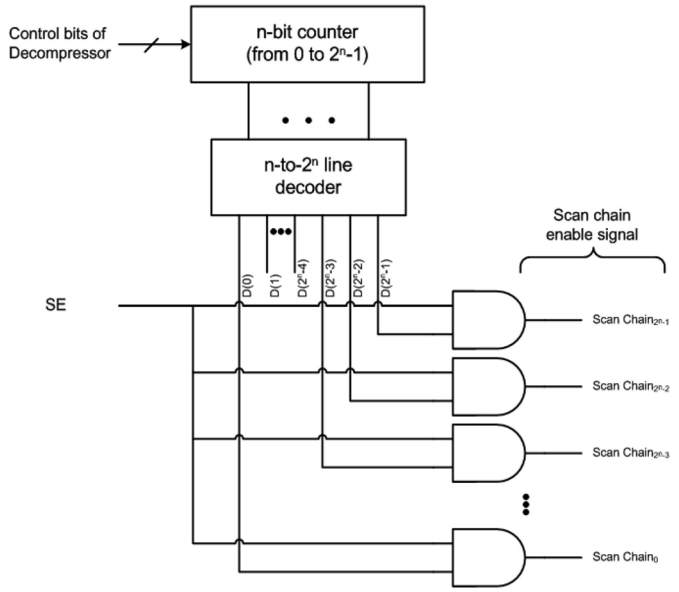

Fig. 10. Chain selector for multiple scan chains.

Consider the following example. If the requested split data match the R-TRC data completely, these data can be served to the scan cells by using the feedback mode of the R-TRC without any insertion data. However, a few data are inserted into the R-TRC using the twist mode for making next data if the requested split data match the R-TRC data partially. Next, the R-TRC data are transmitted to the scan cells by using feedback mode. Hence, the lower twist mode requests that can be served from the R-TRC, the lower the amount of the inserted data from the ATE becomes.

In order to use this module efficiently, CGU has an important role when controlling its clock and selecting its mode. Fortunately, the CGU operation is not complicated as mentioned above in Section IV-B. Finally, this approach can be used for reducing the test time and improving the compression ratio in spite of a lower hardware area overhead compared to that of the dictionary method and CSR.

D. Chain Selector

In recent industrial design trends, the multiple scan chain is preferred to a serial single scan chain because a serial scan chain cannot ensure patterns in all of the subspaces spanned by the inputs of all of the output logic cones that are exhaustively generated [39]. Unfortunately, most existing code-based methods cannot exploit the existence of multiple scan chains in a core [40]. For application of the multiple scan chains by the proposed architecture, the chain is given in Fig. 10. The chain selector can send an enable signal to only one of the multiple scan chains properly. It is composed of a demultiplexer and counter and this selector can be controlled by the control bits of the decompressor. When SE is 1, a selected scan chain receives this enable signal according to the output of the counter, but the other chains receive the disable signal, 0. Except for the scan shift and capture mode, all enable signals of the multiple scan chains are assigned 0, because all outputs are disabled when SE is set to 0.

TABLE VII
COMPRESSION RATIO WITH DIFFERENT l_{SR}

Circuit	# of Scan Cells	Compression Ratio (%)				
		l_{SR}				Max
		8	16	24	32	
s5378	214	72.11	74.75	74.83	**75.39**	**75.39**
s9234	247	70.00	**72.49**	71.30	71.22	**72.49**
s13207	700	80.91	79.80	86.94	**87.39**	**87.39**
s15850	611	72.27	78.23	**79.11**	78.46	**79.11**
s38417	1664	72.54	75.21	75.99	**76.65**	**76.65**
s35854	1464	75.54	79.45	79.69	**79.72**	**79.72**
b17	1452	83.49	88.73	90.03	**90.70**	**90.70**
b18	3356	85.02	90.59	92.33	**93.10**	**93.10**
Average		76.11	79.91	81.27	81.58	81.82

V. EXPERIMENTAL RESULTS

To examine the improved effects of the proposed method, experiments are performed on the six ISCAS'89 benchmark circuits and two large International Test Conference (ITC)'99 benchmark circuits [27]. All test data are generated from TetraMAX which is the test generation tool of Synopsys with the dynamic compaction turned on and random-fill turned off. The proposed method is analyzed from the three main points of view of the test data compression: 1) the compression ratio; 2) the hardware area overhead; and 3) the TAT. These three issues are very important to the industry because they are closely related to the design and the test costs.

First, we present the compression ratio, $C_R(T_D)$, that is estimated with the following equation:

$$C_R(T_D) = \left(\frac{|T_D| - |T_E|}{|T_D|}\right) \times 100 \tag{3}$$

where $|T_D|$ is the size of the original test data and $|T_E|$ is the size of the compressed test data. The compressed test data is composed of both binary and Hi-Z values, and the size of representing Hi-Z values is the same as that for the binary values. In Table VII, the results of the compression ratio with the designated benchmark circuits are presented with four different types of l_{SR}. These results show that the compression ratio generally increases with the increase of l_{SR}, but it is obvious that as l_{SR} increases, the hardware area overhead increases. Clearly, this overhead is linearly dependent on l_{SR}. Therefore, a proper selection of l_{SR} is very important for applying the proposed decompression architecture.

To examine the above relation between l_{SR} and the compression ratio, we recommended using (1) from Section III. Fig. 11 presents the actual compression ratio and evaluated results of the compression ratio from (1) for s38584 circuit. In these results, both lines are saturated at about the same time at the R-TRC length of 12. This saturation point is found to be a suitable l_{SR} because this point results in minimized hardware area overhead although an almost saturated compression ratio is given. Actually, the experimental results show that most of the compression ratios are above 78.3% and below 80.0% since the length of the R-TRC is 12. Hence, a saturated l_{SR} can be obtained from (1) for minimizing the hardware overhead without a heavy reduction in the compression ratio.

TABLE VIII
COMPARISON OF THE COMPRESSION RATIO WITH A VARIETY OF PREVIOUS TEST DATA COMPRESSION METHODS

Circuit	# of Scan Cells	Best Compression Ratio (%)								
		Previous Works								Proposed Method (TSC)
		Golomb [19]	FDR [21]	SHC [15]	VIHC [17]	DIPRL (PR-1) [23]	MHC [40]	FIVO (3-bit) [7]	2^n-PRL [9]	
s5378	214	37.11	48.02	55.10	51.78	60.53	61.07	62.83	54.94	**75.39**
s9234	247	45.25	43.59	54.20	47.25	61.46	59.97	55.17	57.72	**72.49**
s13207	700	79.74	81.30	77.00	83.51	88.66	**89.01**	84.17	88.10	87.39
s15850	611	62.82	66.22	66.00	67.94	75.53	74.91	71.08	74.29	**79.11**
s38417	1664	28.37	43.26	59.00	53.36	58.72	64.36	62.26	58.33	**76.65**
s35854	1464	57.17	60.91	64.10	62.28	75.45	70.97	69.33	72.44	**79.72**
Average		51.74	57.22	62.57	61.02	70.05	70.05	67.47	67.64	**78.46**

TABLE IX
COMPARISON OF THE HARDWARE AREA OVERHEAD WITH A VARIETY OF PREVIOUS TEST DATA COMPRESSION METHODS

Circuit	# of Scan Cells	Hardware Area Overhead (%)						
		Previous Works					Proposed Method (TSC)	
		Golomb [19]	FDR [21]	SHC [15]	VIHC [17]	2^n-PRL [9]	$l_{SR} = 8$	$l_{SR} = 16$
s5378	214	4.0	7.5	16	5.8	4.6	5.6	8.0
s9234	247	3.2	5.5	13	4.6	3.6	4.4	6.3
s13207	700	4.1	3.1	5.7	2.2	1.7	1.9	2.8
s15850	611	2.0	3.2	6.5	2.3	1.8	1.8	2.7
s38417	1664	0.5	1.1	2.0	0.7	0.6	0.8	1.2
s35854	1464	0.7	1.1	2.0	0.7	0.6	0.7	1.0

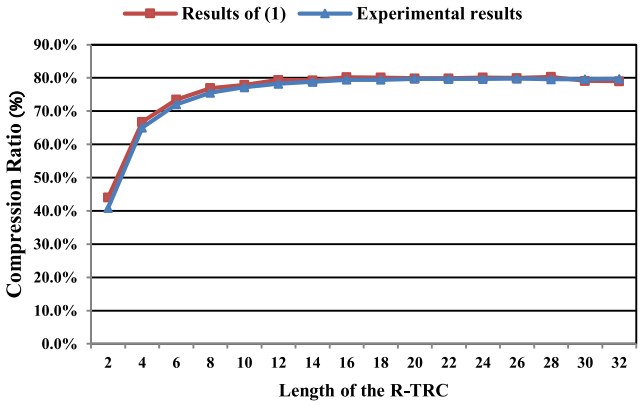

Fig. 11. Correlation between the evaluations and actual experimental results.

Table VIII compares the compression ratio with a variety of previous test data compression methods: 1) Golomb; 2) FDR; 3) selective Huffman coding; 4) variable-length input Huffman coding (VIHC); 5) DIPRL; 6) multilevel Huffman coding (MHC); 7) fixed input variable output (FIVO); and 8) 2^n-PRL. The highest compression ratio is illustrated in bold. Except for the s13027 circuit, the proposed method covers the highest compression ratios (above 5% up to 14%) compared to the highest compression ratio of the previous work. It can be seen that our proposed method performs up to an average of 11% better than [9].

In our next set of experimental results, we illustrate the hardware area overhead compared to the previous methods in Table IX. Note that the hardware area overhead of the MHC and FIVO methods is too large, so they are not written in the comparison table. To acquire this overhead, the proposed decompressor, the R-TRC and the ISCAS'89 circuit are synthesized with a different l_{SR} using Synopsys' design compiler. Note that, the overhead of the tri-state detector is not combined with this area overhead because it can consists of only six transistors. The features of the proposed architecture are slightly dependent on the size of the circuit; hence the overhead is negligible with the large circuits. This feature is shown in the last column of Table IX and its overhead is below 1%. When l_{SR} is 8, this overhead is mostly situated between VIHC and 2^n-PRL and the compression ratio is higher in the case of the s5378, s9234, s38417, and s38584, but the other circuits are lower than 2^n-PRL. When l_{SR} is 16, the hardware area overhead is a little higher than VIHC and 2^n-PRL in the case of the small circuits such as s5378 and s9234. However, in the other cases such as s13207, s15850, s38417, and s35854, the difference in this overhead is below 1.1%, hence it is negligible.

The International Technology Roadmap for Semiconductors 2012 report [41] states that recent industrial circuits increase the size of logic exponentially, i.e., the number of logic gates is above 501 million from 2014 onward; this number is rising at more than 100 million gates a year. Therefore, the hardware area overhead of the proposed method can be predicted to below 1.0% and this method is more suitable for effectively applying these industrial circuits.

In our experimental results, we estimate the comparison of the TAT with a variety of previous test data compression methods and with different frequency ratios (α). The α is the ratio between the chip internal clock (f_{chip}) and the ATE operating clock (f_{ATE}); it is presented as follows: $\alpha = f_{\mathrm{chip}}/f_{\mathrm{ATE}}$.

TABLE X
COMPARISON OF THE TAT WITH THE DIFFERENT FREQUENCY RATIOS

Circuit	# of Scan Cells	TAT (ATE Clock Cycles), l_{SR} = 16					
		Method	α=2	α=4	α=6	α=8	α=32
s5378	214	FDR [21]	24933	16803	15259	14039	-
		VIHC [17]	**15868**	11569	10777	10137	-
		2^n-PRL [9]	17346	13222	12393	11453	10877
		TSC	18057	**11499**	**9797**	**8145**	**6493**
s9234	247	FDR [21]	42066	29206	26675	24086	-
		VIHC [17]	24895	17994	16905	16905	-
		2^n-PRL [9]	27354	20216	19037	17479	16591
		TSC	**23886**	**15438**	**13326**	**11214**	**9102**
s13207	700	FDR [21]	116101	70361	57089	48538	-
		VIHC [17]	89865	52769	44180	36065	-
		2^n-PRL [9]	94924	55844	44632	36306	20490
		TSC	**59555**	**33859**	**27435**	**21011**	**14587**
s15850	611	FDR [21]	65020	42270	36732	32362	-
		VIHC [17]	**47366**	32513	29437	26692	-
		2^n-PRL [9]	48274	32120	28278	24511	19791
		TSC	47620	**29368**	**24805**	**20242**	**15679**
s38417	1664	FDR [21]	186261	123700	113451	10521	-
		VIHC [17]	**104642**	**74293**	**67940**	**62625**	-
		2^n-PRL [9]	115031	86671	80638	73454	68638
		TSC	141548	89964	77068	64172	**51276**
s35854	1464	FDR [21]	179530	118628	104630	93260	-
		VIHC [17]	**126969**	92632	83130	82582	-
		2^n-PRL [9]	128566	86694	76304	66597	54869
		TSC	129586	**79170**	**66566**	**53962**	**41358**

As most of the test compression methods are used for the at-speed scan test, both clocks require synchronization. Therefore, the TAT and proposed method are affected by the frequency ratio.

In our method, the TAT is estimated with the following equation:

$$\text{TAT(cycle)} = T_{\text{code}} + T_{\text{oper\_shift}} \tag{4}$$

where T_{code} is the time that the compressed test data are inserted in the test chip according to the ATE operating clock and $T_{\text{oper\_shift}}$ is the ATE waiting clock for transmitting the R-TRC values to the scan cells. The TAT is estimated as above (4), and it is compared to the previous existing test data compression method in Table X. Here, the methods that are specified their TATs unclearly are not selected for the comparison targets. Except for a few of the cases, the proposed method outperforms most of the cases. To be more specific, the proposed method is reduced to 20%–45% by comparing it to 2^n-PRL with an α of 32.

It is important that whether the proposed method can be applied to the real design with high-compression ratio. If the compression ratio is about 70%, it is hard to use the proposed method for industrial circuit. For this reason, the additional experiments are performed to apply some real designs and these results are shown in Table XI. These real designs are composed of 5.8–8.4 M gates and 380–420 K scan cells. The results show that it can obtain 64-72X compression ratio and these ratios are much higher than the results of the benchmark circuits. Hence, the proposed method can improve the compression ratio in proportion to the design size. These results show that the proposed method can be applied to the real-industrial design.

TABLE XI
INFORMATION OF THE REAL-DESIGN CIRCUIT AND COMPRESSION RATIO

Circuit	# of Scan Cells	# of Gates	Test Coverage (%)	# of Patterns	CR
CKT-1	387,578	5,841,273	97.37	30047	64.69X
CKT-2	420,735	8,472,580	97.84	30050	72.13X

VI. CONCLUSION

In this paper, we present a new input test data compression method called TSC and a decompression architecture with a tri-state detection circuit. Our proposed method is more effective concerning three main factors: 1) the compression ratio; 2) the hardware area overhead; and 3) the TAT. Experimental results show that in most of the cases, the performance of the proposed method is more outstanding than the results of previous methods. This is especially the case with circuits larger than the ISCAS'89 benchmark circuits such as the ITC'99 benchmark circuit which also acquired a high-compression ratio above 90%. To conclude, the proposed method's compression ratio improved up to an average of 8.41% compared to the best previous results and the TAT was reduced up to approximately 17 000 cycles without high-hardware area overhead.

ACKNOWLEDGMENT

The authors would like to thank the SoC Design Technology Team, System IC Research and Development Laboratory, LG Electronics, Inc., for providing the test data of industrial circuits.

REFERENCES

[1] N. Touba, "Survey of test vector compression techniques," *IEEE Des. Test Comput.*, vol. 23, no. 4, pp. 294–303, Aug. 2006.

[2] Y. Yu, Z. Yang, and X. Peng, "Test data compression based on variable prefix dual-run-length code," in *Proc. Instrum. Meas. Technol. Conf.*, Graz, Austria, May 2012, pp. 2537–2542.

[3] J. Shao and J. Ding, "Research on VLSI test compression," in *Proc. Int. Conf. Comput. Sci. Netw. Technol.*, Harbin, China, Dec. 2011, pp. 545–548.

[4] M. Chloupek and O. Novak, "Test pattern compression based on pattern overlapping and broadcasting," in *Proc. 10th Int. Workshop Electron. Control Meas. Signals*, Liberec, Czech Republic, Jun. 2011, pp. 1–5.

[5] Z. Wang, H. Fang, and K. Chakrabarty, "Deviation-based LFSR reseeding for test-data compression," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, no. 2, pp. 259–271, Feb. 2009.

[6] S. Wang, W. Wei, and Z. Wang, "A low overhead high test compression technique using pattern clustering with n-detection test support," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 12, pp. 1672–1685, Dec. 2010.

[7] C. Y. Lin, H. C. Lin, and H. M. Chen, "On reducing test power and test volume by selective pattern compression schemes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 8, pp. 1220–1224, Aug. 2010.

[8] S. Wang and W. Wei, "Cost efficient methods to improve performance of broadcast scan," in *Proc. IEEE Asian Test Symp.*, Sapporo, Japan, Nov. 2008, pp. 163–169.

[9] L. J. Lee, W. D. Tseng, R. B. Lin, and C. H. Chang, "2^n pattern run-length for test data compression," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 4, pp. 644–648, Apr. 2012.

[10] K. Basu and P. Mishra, "Test data compression using efficient bitmask and dictionary selection methods," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 9, pp. 1277–1286, Sep. 2010.

[11] P. Sismanoglou and D. Nikolos, "Input test data compression based on the reuse of parts of dictionary entries: Static and dynamic approaches," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 11, pp. 1762–1775, Apr. 2012.

[12] A. J. Ahne, "Tri-state detection circuit for use in devices associated with an imaging system," U.S. Patent 7 259 588, Aug. 21, 2007.

[13] D. A. Huffman, "A method for the construction of minimum redundancy codes," *Proc. IRE*, vol. 40, no. 9, pp. 1098–1101, Sep. 1952.

[14] A. Jas, J. Ghosh-Dastidar, and N. A. Touba, "Scan vector compression/decompression using statistical coding," in *Proc. IEEE VLSI Test Symp.*, Dana Point, CA, USA, May 1999, pp. 114–120.

[15] A. Jas, J. Ghosh-Dastidar, M.-E. Ng, and N. A. Touba, "An efficient test vector compression scheme using selective Huffman coding," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 6, pp. 797–806, Jun. 2003.

[16] X. Kavousianos, E. Kalligeros, and D. Nikolos, "Optimal selective Huffman coding for test-data compression," *IEEE Trans. Comput.*, vol. 56, no. 8, pp. 1146–1152, Aug. 2007.

[17] P. T. Gonciari, B. M. Al-Hashimi, and N. Nicolici, "Variable-length input Huffman coding for system-on-a-chip test," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 6, pp. 783–796, Jun. 2003.

[18] C. Giri, "Test data compression by split-VIHC (SVIHC)," in *Proc. Int. Conf. Comput. Theory Appl. (ICCTA)*, Kolkata, India, Mar. 2007, pp. 146–150.

[19] A. Chandra and K. Chakrabarty, "System-on-a-chip test-data compression and decompression architectures based on Golomb codes," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 20, no. 3, pp. 335–368, Mar. 2001.

[20] A. Jas and N. Touba, "Test vector decompression via cyclical scan chains and its application to testing core-based designs," in *Proc. IEEE Int. Test Conf.*, Washington, DC, USA, Oct. 1998, pp. 458–464.

[21] A. Chandra and K. Chakrabarty, "Test data compression and test resource partitioning for system-on-a-chip using frequency-directed run-length (FDR) codes," *IEEE Trans. Comput.*, vol. 52, no. 8, pp. 1076–1088, Aug. 2003.

[22] A. H. El-Maleh, "Test data compression for system-on-a-chip using extended frequency-directed run-length (EFDR) code," *IET Comput. Digit. Tech.*, vol. 2, no. 3, pp. 155–163, May 2008.

[23] X. Ruan and R. S. Katti, "Data-independent pattern run-length compression for testing embedded cores in SoCs," *IEEE Trans. Comput.*, vol. 56, no. 4, pp. 545–556, Apr. 2007.

[24] L. Li, K. Chakrabarty, and N. Touba, "Test data compression using dictionaries with selective entries and fixed-length indices," *ACM Trans. Design Autom. Electron. Syst.*, vol. 8, no. 4, pp. 470–490, Oct. 2003.

[25] S. M. Reddy, K. Miyase, S. Kalihara, and I. Pomeranz, "On test data volume reduction for multiple scan chain design," in *Proc. VLSI Test Symp.*, Monterey, CA, USA, 2002, pp. 103–108.

[26] F. Wolff and C. Papachristou, "Multiscan-based test compression and hardware decompression using LZ77," in *Proc. Int. Test Conf.*, Baltimore, MD, USA, 2002, pp. 331–339.

[27] T. Nguyen and H. Luong, "3.3 volt CMOS tri-state driver circuit capable of driving common 5 volt line," U.S. Patent 5 467 031, Nov. 14, 1995.

[28] J. Nicolai, "Integrated circuit with mode detection pin for tristate level detection," U.S. Patent 5 198 707, Mar. 30, 1993.

[29] D. Thomson, P. Sheridan, and J. Cleary, "Tri-state input detection circuit," U.S. Patent 6 133 753, Oct. 17, 2000.

[30] *IEEE Standard Test Access Port Boundary Scan Architecture*, IEEE Standard 1149.1-2001, 2001.

[31] (1989). *ISCAS'89 Circuit.* [Online]. Available: http://www.iwls.org/iwls2005/benchmarks.html

[32] T. Hiraide *et al.*, "BIST-aided scan test—A new method for test cost reduction," in *Proc. VLSI Test Symp.*, Napa, CA, USA, 2003, pp. 359–364.

[33] A. Chandra and K. Chakrabarty, "Combining low-power scan testing and test data compression for system-on-a-chip," in *Proc. Design Autom. Conf.*, Las Vegas, NV, USA, Jun. 2001, pp. 166–169.

[34] W. L. Li, P. H. Wu, and J. C. Rau, "Reducing switching activity by test slice difference technique for test volume compression," in *Proc. IEEE Int. Symp. Circuit Syst.*, Taipei, Taiwan, May 2009, pp. 2686–2989.

[35] C. V. Krishna and N. A. Touba, "Reducing test data volume using LFSR reseeding with seed compression," in *Proc. Int. Test Conf.*, Baltimore, MD, USA, 2002, pp. 321–330.

[36] D. Lee and K. Roy, "Viterbi-based efficient test data compression," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 4, pp. 610–619, Apr. 2012.

[37] A. Chandra, K. Chakrabarty, and S. R. Das, "On using twisted-ring counters for testing embedded cores in system-on-a-chip designs," in *Proc. Instrum. Meas. Technol. Conf.*, Budapest, Hungary, May 2001, pp. 216–220.

[38] B. Zhou, Y.-Z. Ye, and Y.-S. Wang, "Simultaneous reduction in test data volume and test time for TRC-reseeding," in *Proc. 17th Great Lakes Symp. VLSI*, Stresa, Italy, Mar. 2007, pp. 49–54.

[39] Z. Zhang and R. D. McLeod, "An efficient multiple scan chain testing scheme," in *Proc. 6th Great Lakes Symp. VLSI*, Ames, IA, USA, Mar. 1996, pp. 294–297.

[40] X. Kavousianos, E. Kalligeros, and D. Nikolos, "Multilevel-Huffman test-data compression for IP cores with multiple scan chains," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 16, no. 7, pp. 926–931, Jul. 2008.

[41] ITRS. (2012). *Edition Reports*. [Online]. Available: http://www.itrs.net

Sungyoul Seo received the B.S. degree in electronic engineering from Kwangwoon University, Seoul, Korea, in 2013. He is currently pursuing the combined Ph.D. degree with the Department of Electrical and Electronic Engineering, Yonsei University, Seoul.

His current research interests include design-for-testability, scan-based testing, test data compression, and low-power testing.

Yong Lee received the B.S. and M.S. degrees in electrical and electronic engineering from Yonsei University, Seoul, Korea, in 2005, where he is currently pursuing the Ph.D. degree.

He was an Engineer with the System IC Business Team, LG Electronics, Seoul. His current research interests include design-for-testability, design flow, verification, and validation.

Sungho Kang (M'89) received the B.S. degree from Seoul National University, Seoul, Korea, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Texas at Austin, Austin, TX, USA, in 1992.

He was a Research Scientist with the Schlumberger Laboratory for Computer Science, Schlumberger, Inc., Houston, TX, USA, and a Senior Staff Engineer with the Semiconductor Systems Design Technology, Motorola, Inc., Schaumburg, IL, USA. Since 1994, he has been a Professor with the Department of Electrical and Electronic Engineering, Yonsei University, Seoul. His current research interests include very large-scale integration/system-on-chip design and testing, design-for-testability, built-in self-test, defect diagnosis, and design-for-manufacturability.