A Reconfigured Twisted Ring Counter Using Tristate Coding For Test Data Compression

[1]R.Kanagavalli, [2]Dr.O.Saraniya

[1]PG Scholar, [2]Assistant Professor
Department of Electronics and Communication Engineering,
Government College of Technology, Coimbatore, India

Abstract— **Testing plays an important role in VLSI. Attaining a high-test quality in SOC requires more test patterns targeting delay faults and other fault models beyond stuck-at faults. One of the biggest challenges in the testing industry is increasing test data volume. As the test data volume increases, memory modification of automatic test equipment (ATE) is required and additional test application time (TAT) is required. As a result, the cost to sufficiently test SOCs increases. To overcome these problems test data compression method is used. In this project, reconfigured twisted ring counter(R-TRC) using tri-state coding was used for test data compression. This method is based on reusing a stored set of test data with a twisted ring counter. For improving the compression efficiency, tri-state coding (TSC) is used in this compression method.**

IndexTerms— **Tri state coding (TSC), Reconfigured Twisted ring counter(R-TRC), Automatic test equipment (ATE).**

I. Introduction

In System on chip (SOCs) more intellectual property cores and modules can be integrated into the single chip due to the technology process scale up. According to this the size of the test data volume increases, the memory modification is required in the Automatic test equipment. In testing industry the test data volume is the biggest challenge. To overcome these problem test data compression method is used.

Test data Compression is the most preferred test method in industry. It is suitable to the scan testing and compatible with conventional design rules. Test data compression method reduces the amount of test data required for external tester such as ATE (Automatic Test Equipment). Compression schemes are stored at ATE and test data inputs are transmitted to CUT through test ports.

The advantage of test data compression is that it generates the complete set of patterns applied to the CUT with ATPG, and this set of test patterns are optimizable with respect to the desired fault coverage. Test data compression is also easier to adopt in industry because it’s compatible with the conventional design rules and test generation flows for scan testing. Test data compression provides two benefits.

1) First, it reduces the amount of data stored on the tester, which can extend the life of older testers that have limited memory.
2) Second, this is the more important benefit, which applies even for testers with plenty of memory. It can reduce the test time for a given test data bandwidth.[2]

The decompressor expands the data from n tester channels to fill greater than n scan chains. Increasing the number of scan chains shortens each scan chain, in turn reducing the number of clock cycles needed to shift in each test vector. Test data compression method must compress the test vectors losslessly that is, it must reproduce all the specified bits after decompression to preserve fault coverage. In lossy compaction the output response, on the other hand, can use which does not reproduce all data, losing information with negligible impact on fault coverage. Ideally, the output response could be compressed using just a multiple-input signature register (MISR)[3]. Nondeterministic values in the output response would corrupt the final signature. Researchers have developed several schemes to address the problem of unspecified values in the output response, including eliminating the source of the unknown values, selectively masking the unknown values in the output stream, or using an output compression scheme that can tolerate the unknown values.

Test vectors are highly compressible because typically only 1% to 5% of their bits are specified (care) bits. The rest are unspecified bits (don’t-cares), which can take on any value with no impact on the fault coverage [1]. A test cube is a deterministic test vector in which the bits that ATPG does not randomly fill the don’t-cares. In addition to containing a very high percentage of don’t-cares, test cubes also tend to be highly correlated because faults are structurally related in the circuit. These factors are exploitable to achieve high amounts of compression.

II. Test Data Compression Techniques

Test data compression schemes fall broadly into three categories:

1) Code based schemes
2) Linear decompression based schemes
3) Broadcast scan based schemes

Linear decompression based schemes generates a test cube using LFSR (Linear Feedback Shift Register) or by using simple XOR circuit. In broadcast scan based schemes, tester broadcast few control bits and generates a large number of test bits to scan chains. The linear decompression and broadcast scan based schemes can give the better compression efficiency. But it needs additional Automatic test pattern generator and fault simulation. Automatic test pattern generator generates more test patterns than the code based test data compression scheme.[2] These methods have been required large power consumption because the test patterns of ATPG have many switching activities and it degrades the testing reliability.

A. Code based test data compression

Code based schemes compresses the test data using a number of code words which are made up of binary bit streams. It reduces the required test data and decrease the testing time with smaller hardware in comparison to other methods[2]. This method consists of on-chip decoder and it decodes the compressed test data from the Automatic test equipment, delivers the decompressed data to scan cells. This on-chip decoder can be easily designed with simple structure that consists of small Finite state machine and some controllers. The major issues in SOCs are the amount of test data required for testing and the test application time. Code based test data compression method is the best solution for these problems. Code based schemes encodes the test cubes which consist of test data with unspecified bits.

Huffman coding is the most effective statistical based coding method. It provides the shortest average code word length and high compression ratio. The drawback in the Huffman coding method is that it requires large internal hardware. Run length based compression method encodes the runs of patterns as values and counts. The examples of run length based compression schemes are Golomb coding[4], Cyclical scan register based run length code, Frequency directed run length code, Extended frequency directed run length code, Dual run length code and Data independent pattern run length code. These run length based compression schemes can improve the compression efficiency. The major drawback of run length based compression method is that it requires large hardware area overhead.

2^n PRL compression method encodes the $2^{|n|}$ runs of compatible or inversely compatible test patterns. The advantage of this method is that it has simple and easy decompression logic.[10] But it does not have sufficient compression efficiency.

Dictionary based code compression is the fast compression mechanism and it has a good compression efficiency. It is a new method based on reuse parts of dictionary entries to improve the compression efficiency. The major drawback of dictionary based compression is that it requires large amount of memory for storing the dictionary patterns and it requires hardware area overhead.

B. TSC based test data compression

Tri State Coding (TSC) based test data compression method based on reusing a stored set with R-TRC (Reconfigured Twisted Ring Counter). For improving the compression efficiency, tri-state coding (TSC) is used in this compression method. The R-TRC is consists of a ring counter and a multiplexer (MUX). The MUX is used to enables both feedback and twist mode and the twisted data can be selected as Cin data. Based on the MUX selection this module can save and change the internal data[6]. The compression procedure following the TSC is that, the test patterns are extracted from the ATPG and it consists of unspecified bits. 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, fully 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.

III. Compression Algorithm

The flow of data compression process is shown in the Fig.1. It consists of test data input, and split the test data by the length of reconfigured twisted ring counter. The splitted data consist of unspecified bits which are replaced by binary values using the X-filling algorithm. Using tri-state coding the test data is compressed.

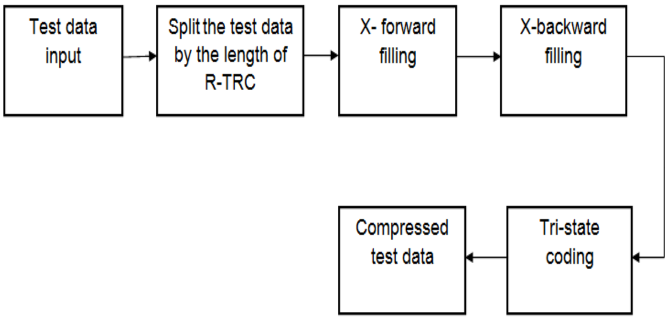


Fig.1 Data compression process

A. Test data input

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%, 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[1].

Input 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, it has been observed that the test data contain a number of unspecified bits. The example test data input taken here is

D1=0xxx010xxxx01x0xxxxxx11xxx0xxx
D2=10xxx0xxxxxxxx00x10100xx1xxxxx

B. *Split test data*

The input test data is splited into the length of the R-TRC (Reconfigured twisted ring counter) which is used to reuse the stored test data set and it improves the compression efficiency. The R-TRC is composed of a ring counter and a multiplexer (MUX) as shown in Fig.2

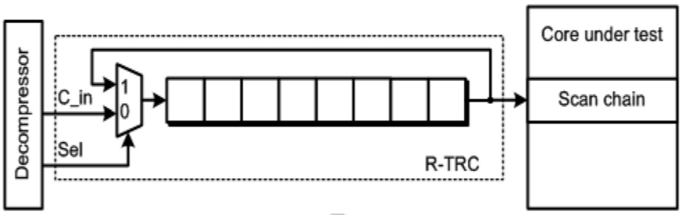
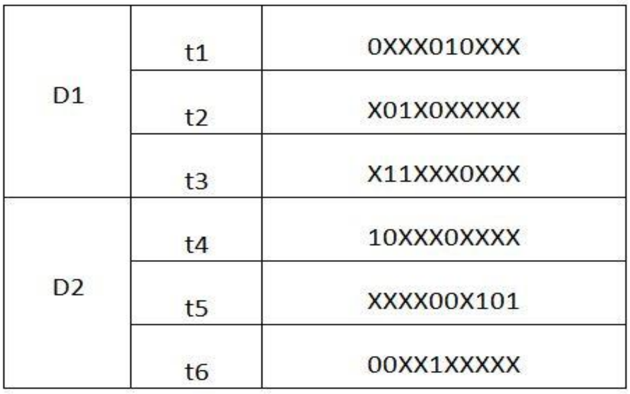


Fig.2 Simple test architecture using the 10-bit R-TRC.

The original test data set, *D*, for a circuit with *n* test patterns, which are generated by the ATPG, $D = \{D1 , D2 , ..., Dn \}$, where the length of *i*th test data, *Di* , which consists of 0s, 1s, and Xs, match the length of the scan chain, *l*SC. After acquiring the above test data, *D*, the next strategy is for each *Di* to be split into the length of the R-TRC, *l*SR. Hence, this makes the $m \times n$ split test data sets, $Di = \{T(i{-}1)m{+}1 , T(i{-}1)m{+}2 , . . . , Tim \}$, where each *Ti* is a subset of *Di* and the length of these subsets is *l*SR. Hence, test data *D* is composed of the subsets of *Ti* with slices of $m \times n$,

The test data input is splited by the length of twisted ring counter (lsc). Let us assume that lsc is taken as 10 bit. So 30 bit input data D1 and D2 are splited by 10 bits,

TABLE I
SPLIT TEST DATA



D1	t1	0XXX010XXX
	t2	X01X0XXXXX
	t3	X11XXX0XXX
D2	t4	10XXX0XXXX
	t5	XXXX00X101
	t6	00XX1XXXXX

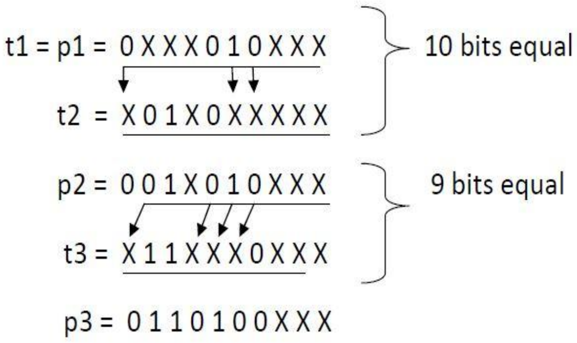
C. *X-filling algorithm*

X-filling algorithm is used to replace the unspecified bits by binary value '0' and '1'. 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

a) Forward X-filling
b) Backward X-filling

a. *Forward X-filling algorithm*

The unfilled split test data, *Ti*, are converted to forward X-filled split data, *Pi*, where $Pi = \{P(i{-}1)m{+}1 , P(i{-}1)m{+}2 , ..., Pim \}$; they are a subset of *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 it is used in the next step for encoding the data. *l*SC is 30 and *l*SR is 10, the forward X-filling method is represented in Fig. 5.5, where *xi* indicates a binary or an unspecified bit.



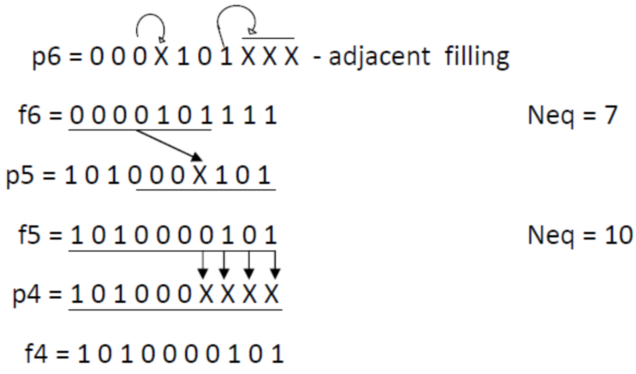
This method is preceded in the forward direction following the arrow. If it is a first test data, X-filling is not applied to this data and *N*eq is assigned 0. From the second test data to the last test data, searching the matching part between the current split test data and the previous one is conducted 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.

TABLE II
FORWARD X-FILLING

	Partially filled data	Neq
p1	0XXX010XXX	0
p2	001X010XXX	10
p3	0110100XXX	9
p4	101000XXXX	8
p5	101000X101	10
p6	000X101XXX	7

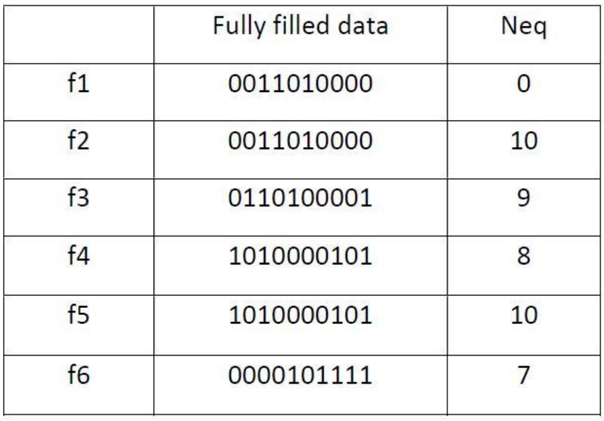
b. Backward X-filling algorithm

In contrast to a previous step, the order of the next X-filling is backward; hence this process is performed from the last to first data. To be more specific, *Pi* is converted to the backward X-filled split test data, *Fi*, and *Fi* is encoded to the final encoded split test data, *Ei*, where *Fi* = {*F(i*−*1)m*+1 , *F(i*−*1)m*+2 , . . . , *Fim* } is a subset of *F*. This method is the procedure for using the R-TRC.



First of all, the last data, P6, is applied to the adjacent filling, which is assigned a previous bit if there is an unspecified bit. Next, the content between F5 [lSR − N6eq + 1] and F5 [lSR] is assigned, in order, to the content from F6[1] to F6 [N6eq]. On the other hand, the other contents are still applied to the adjust filling. When this procedure reaches F1 , the full X-filling test data, TF, can be acquired.

TABLE III
BACKWARD X-FILLING



	Fully filled data	Neq
f1	0011010000	0
f2	0011010000	10
f3	0110100001	9
f4	1010000101	8
f5	1010000101	10
f6	0000101111	7

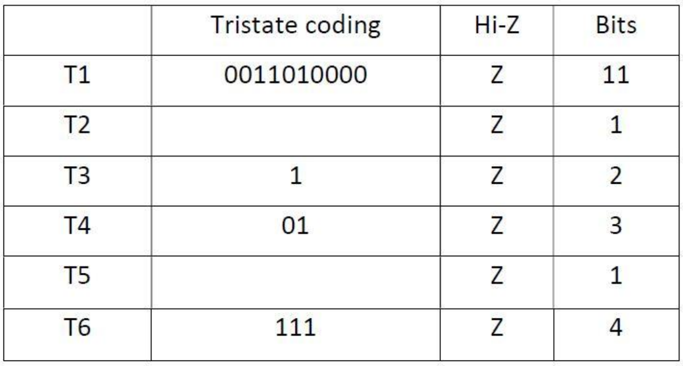
D. Tristate coding

TSC is used to reduce the internal memory required to store the test data and it is used to improve the compression efficiency. 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 High impedance value transmission from its connected ports, the formal test data compression methods have never attempted to use the High impedance values in the design-for-testability (DFT) field[5]. However, we initially use the High impedance value for improving the compression efficiency that is beneficial to the environment-limited channel bandwidth. Hence, entropy can be increased for this situation.

Tri-state coding for the DFT application is very useful, especially in case of test data compression method with a dependence on the ATE. For this, there is no additional cost for modifying the external tester because most existing ATE generally supports high impedance output. Besides, it overcomes the restriction in the number of TDI ports. 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.

The final step is that the results so far, *Fi*, are encoded to TSC, *Ei*, using the R-TRC, where *Ei* = {*E(i*−1*)m*+1 , *E(i*−1*)m*+2 , . . . , *Eim* } is a subset of *E*.[1] Here, *Ei* has variable length and it can be found to be *l*SR−*Ni*eq +1 and this content consists of the R-TRC insertion values from *Fi* [*Ni*eq + 1] to *Fi* [*l*SR] and a high impedance signal.

TABLE IV
TRISTATE CODING

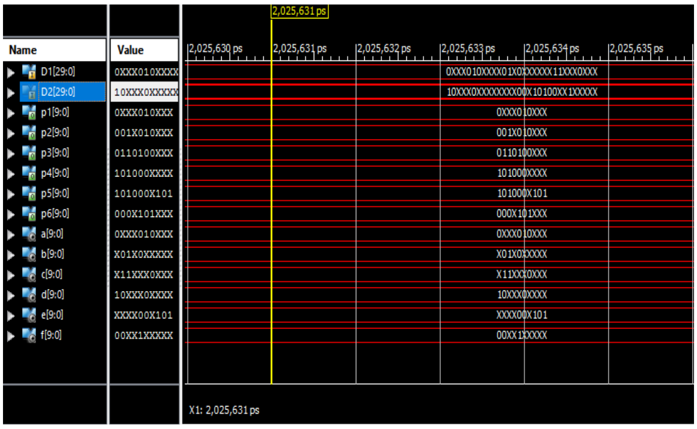


	Tristate coding	Hi-Z	Bits
T1	0011010000	Z	11
T2		Z	1
T3	1	Z	2
T4	01	Z	3
T5		Z	1
T6	111	Z	4

First of all, the first encoded data, *E1* , is composed of *F1* and a High impedance value. However, the next *E2* has only a High impedance because the next *F2* is exactly equal to *F1* . In case of *E3* , it is composed of 1 and a High impedance value. It means that if 1 is inserted to the R-TRC, it can make *F3* data because the R-TRC is stored to *F2* data. In Table IV, the third column data is essentially inserted contents to make the test data. This procedure is iterated until *Ei* reaches the last encoded test data. As a result, the final compressed data size is 22-bit, which is composed of binary and High impedance values.

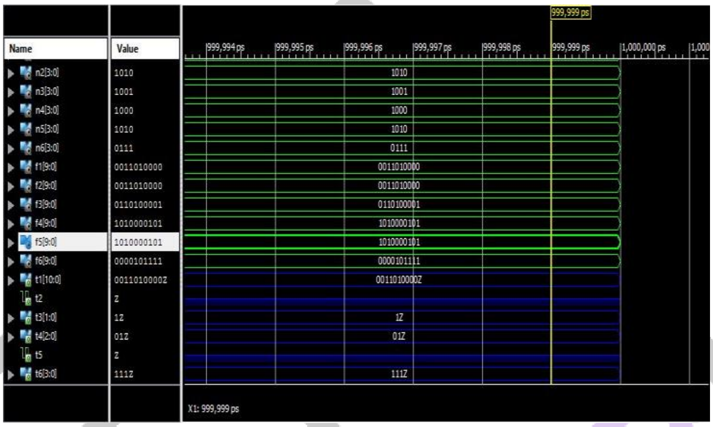
IV. RESULTS

The test data input is applied to the test data compression algorithm that is shown in the Fig.3 shows. Input has 30 bits which has the unspecified values ‘x’. These 30 bits are splitted into 10 bits a, b, c, d, e, f shown in figure and the unspecified are partially replaced by forward X-filling algorithm. Fig.4 describes the Backward X-filling algorithm and compressed output by the tri-state coding. The remaining unspecified bits are replaced using the backward X-filling algorithm and it is compressed using tri-state coding. The compressed output has the dynamic size and it consist of high impedance value.



Name	Value
D1[29:0]	0XXX010XXXX
D2[29:0]	10XXX0XXXXX
p1[9:0]	0XXX010XXX
p2[9:0]	001X010XXX
p3[9:0]	0110100XXX
p4[9:0]	101000XXXX
p5[9:0]	101000X101
p6[9:0]	000X101XXX
a[9:0]	0XXX010XXX
b[9:0]	X01X0XXXXX
c[9:0]	X11XXX0XXX
d[9:0]	10XXX0XXXX
e[9:0]	XXXX00X101
f[9:0]	00XX1XXXXX

Fig.3. partially filled output



Name	Value
n2[3:0]	1010
n3[3:0]	1001
n4[3:0]	1000
n5[3:0]	1010
n6[3:0]	0111
f1[9:0]	0011010000
f2[9:0]	0011010000
f3[9:0]	0110100001
f4[9:0]	1010000101
f5[9:0]	1010000101
f6[9:0]	0000101111
t1[10:0]	00110100002
t2	z
t3[1:0]	1z
t4[2:0]	01z
t5	z
t6[3:0]	111z

Fig.4. TSC output

V. Conclusion

In this project, a new input test data compression method called Tri-state coding with Reconfigured twisted ring counter. Our proposed method is more effective concerning three main factors: 1) the compression ratio; 2) the hardware area overhead; and 3) the TAT. Here the compression ratio is considered. Experimental results show that this method compresses the test data about 63-75%.

References

[1] Sungyoul Seo, Yong Lee, and Sungho Kang, "Tristate coding using reconfiguration of twisted ring counter for test data compression", IEEE trans. on comput-aided design integrat. circuits systs, vol. 35, no. 2, Feb 2016.
[2] N. Touba, "Survey of test vector compression techniques," *IEEE Des. Test Comput.*, vol. 23, no. 4, pp. 294–303, Aug. 2006.
[3] D. Thomson, P. Sheridan, and J. Cleary, "Tri-state input detection circuit," U.S. Patent 6 133 753, Oct. 17, 2000.
[4] 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.
[5] J. Ahne, "Tri-state detection circuit for use in devices associated with an imaging system," U.S. Patent 7 259 588, Aug. 21, 2007.
[6] 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.
[7] 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.
[8] 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.
[9] 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.
[10] 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.
[11] 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.