ELectronics EXpress

LETTER

IEICE Electronics Express, Vol.14, No.13, 1–11

A novel test data compression approach based on bit reversion

Shuo Cai[1a)], **Yinbo Zhou**[2], **Peng Liu**[2], **Fei Yu**[1], **and Wei Wang**[1]

[1] *Hunan Provincial Key Laboratory of Intelligent Processing of Big Data on Transportation & College of Computer and Communication Engineering, Changsha University of Science and Technology, China*

[2] *College of Information Science & Engineering, Hunan University, China*

a) *caishuo@csust.edu.cn*

Abstract: Test data compression is an effective methodology for reducing test data volume and testing time. This paper presents a new test data compression approach based on bit reversion, which compresses data more easier by reversing some test data bits without changing the fault coverage. As there are some don't care bits in test set, when they are filled, many faults will be repeatedly detected with multiple vectors. Correspondingly, a lot of bits in the test set can be modified without affecting the fault coverage. Experimental results show that the proposed method can increase compression ratio of code-based schemes by around 10%.

Keywords: test data compression, code, bit reversion

Classification: Integrated circuits

References

[1] U. S. Mehta, *et al.*: "Run-length-based test data compression techniques: How far from entropy and power bounds? — A survey," VLSI Design **2010** (2010) 670476 (DOI: 10.1155/2010/670476).

[2] P. T. Gonciari, *et al.*: "Improving compression ratio, area overhead and test application time for system-on-a-chip test data compression/decompression," Design, Automation & Test in Europe Conf. (2002) 604 (DOI: 10.1109/DATE.2002.998363).

[3] Z. You, *et al.*: "A scan disabling-based BAST scheme for test cost reduction," IEICE Electron. Express **8** (2011) 1367 (DOI: 10.1587/elex.8.1367).

[4] A. Jas, *et al.*: "An efficient test vector compression scheme using selective Huffman coding," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. **22** (2003) 797 (DOI: 10.1109/TCAD.2003.811452).

[5] X. Kavousianos, *et al.*: "Optimal selective Huffman coding for test-data compression," IEEE Trans. Comput. **56** (2007) 1146 (DOI: 10.1109/TC.2007.1057).

[6] 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. **32** (2013) 1762 (DOI: 10.1109/TCAD.2013.2270433).

[7] T. B. Wu, *et al.*: "Efficient test compression technique for SOC based on block merging and eight coding," J. Electron. Test. **29** (2013) 849 (DOI: 10.1007/s10836-013-5415-7).

DOI: 10.1587/elex.14.20170502
Received May 14, 2017
Accepted June 5, 2017
Publicized June 21, 2017
Copyedited July 10, 2017

[8] 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. **20** (2001) 355 (DOI: 10.1109/43.913754).
[9] 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. **52** (2003) 1076 (DOI: 10.1109/TC.2003.1223641).
[10] P. T. Gonciari, *et al.*: "Variable-length input Huffman coding for system-on-a-chip test," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. **22** (2003) 783 (DOI: 10.1109/TCAD.2003.811451).
[11] A. H. El-Maleh: "Test data compression for system-on-a-chip using extended frequency-directed run-length code," IET Comput. Digit. Tech. **2** (2008) 155 (DOI: 10.1049/iet-cdt:20070028).
[12] A. Chandra and K. Chakrabarty: "A unified approach to reduce SOC test data volume, scan power and testing time," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. **22** (2003) 352 (DOI: 10.1109/TCAD.2002.807895).
[13] M. Nourani and M. H. Tehranipour: "RL-Huffman encoding for test compression and power reduction in scan application," ACM Trans. Des. Autom. Electron. Syst. **10** (2005) 91 (DOI: 10.1145/1044111.1044117).
[14] I. Hamzaoglu and J. H. Patel: "Test set compaction algorithms for combinational circuits," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. **19** (2000) 957 (DOI: 10.1109/43.856980).
[15] H. K. Lee and D. S. Ha: "HOPE: An efficient parallel fault simulator for synchronous sequential circuits," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. **15** (1996) 1048 (DOI: 10.1109/43.536711).
[16] R. Sankaralingam, *et al.*: "Static compaction techniques to control scan vector power dissipation," Proc.18th IEEE VLSI Test Symposium (2000) 35 (DOI: 10.1109/VTEST.2000.843824).

1 Introduction

With the rapid development of IC (Integrated Circuit) fabrication processing, an increasing number of transistors are integrated into one single chip, however, it causes multiple increase of test data volume, which not only increases the testing time but also exceeds the tester memory capacity [1]. Compressing test data is an effective approach to reduce test data volume.

The goal of test data compression is to reduce the number of binary bits in original test data which can be generated by the test generation tool after circuit design. The compressed test set is stored in the automatic test equipment (ATE) and a decompress circuit on chip is used to decompress the test data and apply it to the circuit under test (CUT), which is shown in Fig. 1 [2]. The decompress circuit is corresponding to the compression approach and is suitable for all the original test set which using this compression approach.

The test data compression approaches can generally be classified into three categories: code-based schemes, linear-decompression-based schemes and broadcast-scan-based schemes [3]. Code-based schemes mainly aim at given test sets, in which original test data are divided into different symbols and each symbol is replaced by a code word to form the compressed data. We don't need to understand

DOI: 10.1587/elex.14.20170502
Received May 14, 2017
Accepted June 5, 2017
Publicized June 21, 2017
Copyedited July 10, 2017

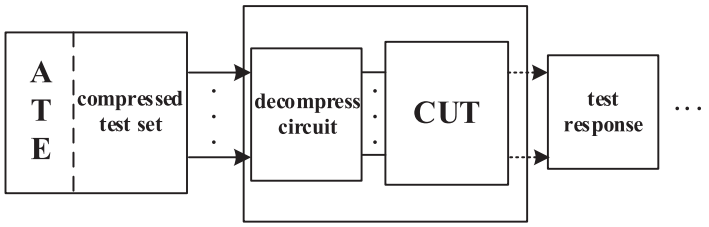

Fig. 1. Test compression structure

the internal structure information of the circuit under test, in addition, the fault simulation and test generation are not needed. Therefore, these schemes are very suitable for test data compression with embedded IP core circuits. According to the differences of symbol division, the coding methods can be classified into two categories. If the scheme encodes a fixed number of input bits, it belongs to the "fixed" category. Similarly, schemes that encode a variable number of input bits are classified under the "variable" category. Huffman coding is the best example of "fixed" scheme. The idea with a Huffman code is to encode symbols that occur more frequently with shorter code words and symbols that occur less frequently with longer code words, such as selective Huffman coding (SHC) [4] and optimal selective Huffman coding (OSHC) [5]. Other "fixed" schemes include dictionary-based coding [6] and block merging coding [7] and so on. Variable categories mainly include two methods: single run-length coding method and double run-length coding method. Single run-length code compression techniques are based on encoding runs of 0 s such as Golomb code [8], frequency-directed run-length (FDR) code [9] and variable input Huffman code (VIHC) [10]. Double run-length code compression techniques are based on encoding both runs of 0 s and runs of 1 s such as extended FDR code (EFDR) [11], alternating run-length coding (AFDR) [12] and mixed double run-length and Huffman coding (RL-HC) [13].

Many bits in test set are don't care bits, which can be filled with any value (0 or 1) without decreasing the fault coverage of CUT. In fact, when don't care bits are filled, many faults will be repeatedly tested by multiple vectors [1]. Based on this, a new approach to improve the compression ratio of test set is proposed in this paper, which reverses some specified bits of test set after the don't care bits are filled. This bit reversion-based approach can make test data easier to be compressed without decreasing the fault coverage. The experimental results indicate that our approach can improve the compression ratio of coding by around 10%.

In this paper, Section 2 introduces some typical coding methods. Section 3 presents the theories and algorithms of test set reversion. Section 4 demonstrates the experimental results. Section 5 concludes the whole paper.

2 Introduction of coding methods

This section introduces three representative coding methods, i.e., FDR code, EFDR code and Optimal SHC, which come from single run-length coding methods, double run-length coding methods and "fixed" schemes respectively.

1) FDR code

Run length refers to the sequence composed by the continuous same symbols. FDR code is based on 0 run length, which means the continuous 0 with a tail 1. In

DOI: 10.1587/elex.14.20170502
Received May 14, 2017
Accepted June 5, 2017
Publicized June 21, 2017
Copyedited July 10, 2017

FDR code, code words are constituted by the prefix and tail of the same length. In group A_i, the length of prefix is i, and the prefix of each group represents the run length of the first code of this group. From Group A_i to A_{i+1}, the length of prefix and tail increases one bit respectively. The size of the ith group is equal to $2i$, that is, group A_i contains $2i$ members. Run length of j is mapped to group A_i, where $i = \lceil \log_2(j+3) \rceil - 1$. The FDR code word is shown in Table I.

Table I. FDR code

Group	Run-length	Group prefix	Tail	Code word
A_1	0	0	0	00
	1		1	01
A_2	2	10	00	1000
	3		01	1001
	4		10	1010
	5		11	1011
A_3	6	110	000	110000
	7		001	110001
	...		...	...
	13		111	110111
...	...	...	...	...

An example of FDR code is illustrated. For test set T = {00000001 1 000000001 000001}, FDR code is applied to obtain T_{FDR} = {110001 00 110010 1011}, which reduces 6 bits. From this example, it can be observed that the efficiency of FDR code is mainly decided by the quantity of 1 in the test set. The fewer the quantity of 1 is, the fewer the test data bits after coding will be.

2) EFDR code

In order to encode both runs of 0 s and 1 s, EFDR code adds one bit based on FDR code to identify the run length is 0 or 1. 0 run length means the continuous 0 with a tail 1, while 1 run length means the continuous 1 with a tail 0. The EFDR code word is shown in Table II.

For test set T = {01 001 1111110 00000001}, EFDR code is utilized to obtain the result of T_{EFDR} = {000 001 11011 0110000}, which reduces 2 bits. From this example, it can be observed that when the binary bit of the test set flips (from 0 to 1 or from 1 to 0), there will be a run length. When the flip number is fewer in a test set, the compression ratio will be higher.

3) Optimal SHC

Optimal SHC belongs to the coding method from fixed length to variable length, which encodes the data blocks of certain length with variable-length code words. Data blocks with high frequency are distributed with short code words and vice versa so as to achieve the purpose of data compression. In order to reduce the hardware cost, data blocks with high frequency apply Huffman coding while the ones with low frequency will not be encoded but introduced by ATE directly.

DOI: 10.1587/elex.14.20170502
Received May 14, 2017
Accepted June 5, 2017
Publicized June 21, 2017
Copyedited July 10, 2017

Table II. EFDR code

Group	Run-length	Group prefix	Tail	Code word runs of 0 s	Code word runs of 1 s
A_1	1	0	0	000	100
	2		1	001	101
A_2	3	10	00	01000	11000
	4		01	01001	11001
	5		10	01010	11010
	6		11	01011	11011
A_3	7	110	000	0110000	1110000
	8		001	0110001	1110001
	...		...	...	...
	14		111	0110111	1110111
...	...	...	...	...	...

The following example will illustrate the process of optimal SHC. In Table III, the test set T is composed by 5 vectors of 16 in length. After dividing the test set T into data blocks of 4 in length, the second column and third column in the table list each category of data blocks and their frequencies. For example, if only the data blocks with the first three highest frequencies are encoded, the frequency of uncoded blocks will be $2/20 + 1/20 = 3/20$. Subsequently, the data blocks with the highest frequencies like 1010, 0000, 1111 and uncoded blocks will apply Huffman coding, which is shown in Fig. 2. The data block 1010 with the highest frequency will be represented with only one bit of 0, while the uncoded blocks need to add the identifying bits of 111 before the data block (such as encoding 0001 into 111 0001), which will be 7 bits. The compression ratio in this case is equal to 38.8%. Therefore, it can be concluded that the fewer the uncoded data blocks in test set are, the higher the compression ratio will be.

Table III. Division of test set and frequency of different blocks

Test set T	Different blocks	Frequency
1010 0000 1010 1111	1010	9/20
1111 0000 1010 0001	0000	5/20
1010 0000 0010 1010	1111	3/20
0000 1010 1010 0000	0001	2/20
1010 1111 1010 0001	0010	1/20

3 Bit reversion theories and algorithms

To understand the bit reversion theories of test set, the concept of test redundancy must be defined first.

Definition 1: In test set T, the quantity of test vectors which can detect a specified fault is called the test redundancy of this fault. The average value of test redundancy of all faults is called as the average test redundancy rate of test set T.

DOI: 10.1587/elex.14.20170502
Received May 14, 2017
Accepted June 5, 2017
Publicized June 21, 2017
Copyedited July 10, 2017

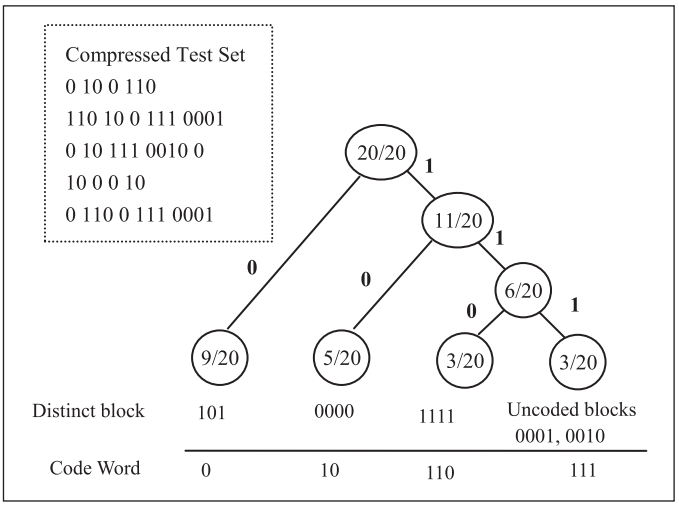

Fig. 2. Optimal SHC

The average test redundancy reflects the degree of a single fault in a circuit being repeatedly detected by test set T. Table IV illustrates the test redundancy rate before and after X (don't care bits) filling.

Table IV. Average test redundancy rate before and after X filling

Circuit	X not filled	X filled with random bits
s5378	8.6	24.9
s9234	8.8	26.4
s13207	7.5	53.8
s15850	7	27.6
s38417	7.5	22.1
s38584	6.2	32.1
Average	7.6	31.2

It can be observed that after X filling in the test set, the test redundancy will increase. Some specified bits in the test set can be reversed without decreasing the fault coverage because of the test redundancy. When the test set changes to benefit the compression, the compression ratio will be improved.

FDR code is taken as the example to illustrate the basic process of test set reversion. As section 2 has mentioned, the purpose of test set reversion is to reduce the number of 1. The algorithm 1 illustrates the basic process of test set reversion: For each 1 s in the test set T, change it to 0 s if the fault coverage does not reduce.

Algorithm 1. TestSetReversion

Coverage ← *FaultCoverage*(T) //solve fault coverage of T
FOR *bit* IN T //bit is a certain bit in T
  IF *bit* = 1 //turn 1 of T into 0
  bit ← 0
    //if the fault coverage is affected, restore
  IF *Coverage* ≠ *FaultCoverage*(T)
      bit ← 1

DOI: 10.1587/elex.14.20170502
Received May 14, 2017
Accepted June 5, 2017
Publicized June 21, 2017
Copyedited July 10, 2017

Factors affecting the test set reversion ability include average redundancy, distribution of test redundancy and bit reversion method. Different reversion strategies are needed for different coding methods. Three general algorithms will be listed as follows.

1) Single run-length reversion algorithm

The purpose of bit reversion for single run-length coding method is to reduce the quantity of 1 s in the test set.

Definition 2: Reverse all the bits 1 of a column in test set successively according to a certain order. If the fault coverage of the whole test set after reversing the bit is not affected, this bit will be reversed. The quantity of reversed bits is called the reversion weight of the column (RWC).

Algorithm 2. SingleRunLengthReversion

```
WHILE T not empty    //when test set T is not empty
/*m column is randomly selected from T to find out the maximum column of
reversion right*/
  MaxColumn ← 0
  MaxWeight ← 0
  FOR i ← 1 to m
    Column = RandomChoose(T)
    IF MaxWeight < weight(Column,0)
      MaxColumn ← Column
      MaxWeight ← weight(Column,0)
/*n types of reversion order is formed randomly to find out the maximum order of
reversion right */
  MaxOrder ← 0
  MaxWeight ← 0
  FOR j ← 1 to n
    Order = RandomOrder()
    IF MaxWeight < weight(MaxColumn, Order)
      MaxOrder ← Order
      MaxWeight ← weight(MaxColumn, Order)
/*find the proper column and reversion order and take this column as this order for
reversion and mark it as reversed.*/
  ReverseColumn(MaxColumn, MaxOrder)
  Remove(MaxColumn)
```

The single run-length reversion algorithm is shown as algorithm 2. Firstly, pick up randomly *m* columns from the test set, and selects among of them the column with the maximal RWC. This column is called the *MaxColumn*. Secondly, for the *MaxColumn* generate randomly *n* different orders and select an order in the way that the maximal RWC can be obtained if the flips are conducted. This order is called the *MaxOrder*. Finally, reverse all the bits 1 for the *MaxColumn* in the order of the *MaxOrder*, and sign subsequently the *MaxColumn* that has been reversed. The three steps are repeated until all the bits 1 in the test set are tried.

2) Double run-length reversion algorithm

The purpose of test set reversion for double run-length coding method is to reduce the quantity of flips. A consecutive sequence of equal bits is called non-flip

DOI: 10.1587/elex.14.20170502
Received May 14, 2017
Accepted June 5, 2017
Publicized June 21, 2017
Copyedited July 10, 2017

sequence. For test set T = {0 111 000 1111111 00000000}, the non-flip sequences are 0, 111, 000, 1111111 and 00000000. The shortest non-flip sequence will be firstly reversed to reduce the flip number, and then T = {1 111 000 1111111 00000000}. If the fault coverage of test set T does not change, we continue to reverse the shortest non-flip sequence at present, get T = {1 111 111 1111111 00000000}. The whole algorithm is shown in algorithm 3.

Algorithm 3. DoubleRunLengthReversion

```
Coverage ← FaultCoverage(T) //solve the fault coverage
FOR i ← 1 to m   //reverse the flip string with length less than i
  FOR NotFlipString IN T // flip string in T
    If length(NotFlipString) < i
      Reverse(NotFlipString) // reverse flip string
        //if the fault coverage is affected, restore
      IF Coverage ≠ FaultCoverage(T)
        Reverse(NotFlipString)
```

3) "Fixed" reversion algorithm

"Fixed" scheme divides test set into blocks of the same fixed length for coding respectively. There are many types of "fixed" scheme, but there is one common point: In coding process, the current data block is compared with the target blocks. If they are equal, a corresponding code word will be assigned to the current block, if not, extra bits will be added in front of the current block to distinguish it from the code words. Therefore, for "fixed" scheme, the purpose of reversion is to make the number of uncoded blocks smaller while applying short code words for coded blocks.

The reversion algorithm is shown in algorithm 4. For "fixed" schemes, the test set T is divided into blocks with fixed length. All of the blocks are grouped into one set, named B. Similarly, the "target blocks" are grouped into a set, named D. For each blocks in B, it will be replaced with one of the target blocks in D, if the replace do not change the fault coverage. The target blocks are updated after every loop. This algorithm is repeated until no blocks can be replaced.

Algorithm 4. BlockReversion

```
Coverage ← FaultCoverage(T) //get the fault coverage of T
D = GetDestBlock()   //get the original target block D
//for each block in B
//replace each block of dest in D without affecting the fault coverage ratio
FOR block IN B
  FOR dest IN D //target block
    block ← dest
    IF Coverage = FaultCoverage(T) //find suitable block
      BREAK
    //if no suitable block is found in D, restore the block
  IF Coverage ≠ FaultCoverage(T)
    recovery(block)
  update(D) // update target block set
```


DOI: 10.1587/elex.14.20170502
Received May 14, 2017
Accepted June 5, 2017
Publicized June 21, 2017
Copyedited July 10, 2017

4 Experimental results

We implemented the bit reversion algorithm using C++ and verified its effectiveness on the ISCAS'89 benchmark circuits. Six largest circuits are selected as the experimental circuits, and the test sets are generated by the Mintest ATPG [14]. Fault simulator "HOPE" [15] is used for fault simulation. Certainly, the test set will be generated by some specific methods during test phase after the design for every actual circuit, and our approach can be suitable for any given original test set. Moreover, bit reversion-based approach has an advantage that requires no new decompress circuit, but uses the existing decompress circuit corresponding to such as FDR, EFDR coding methods and so on. As a result, there is no additional overhead in the process of decompression.

Table V compares the compression ratio changes of three single run-length coding methods before and after reversion (Golomb [8], FDR [9], VIHC [10]). The first column is circuit name, the 2–4 columns are the compression ratio of original test set, the 5–7 columns are the compression ratio of test set after reversion. It can be observed from the table that the compression ratio has improved 7.42%∼11.20% after reversion.

Table V. Compression ratio comparison of single run-length coding methods

Circuit	Original			Test set reversion		
	Golomb	FDR	VIHC	Golomb	FDR	VIHC
s5378	37.11	47.98	51.78	48.63	57.83	60.52
s9234	45.25	43.61	47.25	53.63	50.36	55.06
s13207	79.74	81.3	83.51	83.03	83.40	85.86
s15850	62.82	66.21	67.94	70.55	72.37	74.49
s38417	28.37	43.37	53.36	57.27	62.69	67.19
s38584	57.17	60.93	62.28	64.56	65.60	67.55
Average	51.74	57.23	61.02	**62.94**	**65.38**	**68.44**

As shown in Table VI, after bit reversion, the compression ratio of double run-length reversion has improved 8.42%∼12.07% (AFDR [12], EFDR [11], RL-HC [13]). The double run-length reversion algorithm can better improve the compression ratio of test set than single run-length reversion algorithm. This is because the single run-length reversion algorithm merely fills don't care bits of test set into 0, which makes the redundancy distribution uneven and influences the effects of test set reversion.

Table VII shows the results of Optimal SHC [5] for "fixed" scheme. Compared to the previous two methods, the compression ratio of fixed scheme has improved more by 12.7% with the final compression ratio of 81.22%, exceeding the majority of code-based compression methods.

Finally, we compare test power [16] between original test set and test set after bit reversion. Table VIII gives the average power and peak power of experimental circuits for each method. The single run-length reversion algorithm makes the

DOI: 10.1587/elex.14.20170502
Received May 14, 2017
Accepted June 5, 2017
Publicized June 21, 2017
Copyedited July 10, 2017

Table VI. Compression ratio comparison of double run-length coding methods

Circuit	Original			Test set reversion		
	AFDR	EFDR	RL-HC	AFDR	EFDR	RL-HC
s5378	50.77	53.67	53.75	63.36	63.51	65.89
s9234	44.96	48.66	47.59	62.04	61.34	66.07
s13207	80.23	82.49	82.51	85.91	85.74	88.63
s15850	65.83	68.66	67.34	76.02	75.85	78.65
s38417	60.55	62.02	64.17	69.95	69.49	72.38
s38584	61.13	64.28	62.40	75.24	74.39	78.55
Average	60.58	63.30	62.96	**72.09**	**71.72**	**75.03**

average power consumption and peak power reduce by around 9%, while double run-length reversion algorithm makes those reduce by almost a half. This is because that double run-length reversion algorithm aims to reduce the number of bit flips while the power consumption is produced due to the bit flips of test set. The average power consumption of Optimal SHC increases by 25.73% but the peak power remains almost the same. In fact, the power consumption of "fixed" schemes can be controlled, which needs to select the blocks with few flips as the target blocks, and this will reduce some compression ratio of "fixed" schemes.

Table VII. Compression ratio comparison of "fixed" scheme

Circuit	Original	Test set reversion
	Optimal SHC	Optimal SHC
s5378	59.60	74.94
s9234	60.00	77.58
s13207	85.30	89.75
s15850	72.20	82.77
s38417	64.50	76.77
s38584	69.50	85.52
Average	68.52	**81.22**

Table VIII. Comparison of power consumption

Test set	Single run-length		Double run-length		Optimal SHC	
	Average	Peak	Average	Peak	Average	Peak
Original	58281	207116	38242	177852	**111088**	193153
Reversed	**53088**	**187268**	**22819**	**73660**	139670	**186601**

5 Conclusion

We propose a new test data compression approach based on bit reversion, which makes it easier to compress data by reversing some bits of the test set without decreasing their fault coverage. Although the use of bit reversion-based approach

DOI: 10.1587/elex.14.20170502
Received May 14, 2017
Accepted June 5, 2017
Publicized June 21, 2017
Copyedited July 10, 2017

requires the fault simulation process, a higher compression rate can be achieved. Therefore, more storage space and more test application time can be saved in the process of testing a large number of chips. From this point of view, our approach has very important practical significance.

In order to fully verify the effectiveness of our approach, we propose some general algorithms for three types of coding methods. The experimental results indicate that the proposed approach can greatly improve the compression ratio of test data, and reduce test power consumption to a certain extent.

Besides code-based schemes, linear-decompression-based schemes and broadcast-scan-based schemes can also apply our approach.

Acknowledgments

This paper was supported in part by the National Natural Science Foundation of China (NSFC) under grant No. 61504013 and in part by Hunan Province Undergraduates Innovating Experimentation Project of China under grant No. (2016)283-192.

DOI: 10.1587/elex.14.20170502
Received May 14, 2017
Accepted June 5, 2017
Publicized June 21, 2017
Copyedited July 10, 2017