Hindawi
Security and Communication Networks
Volume 2022, Article ID 6974101, 7 pages
https://doi.org/10.1155/2022/6974101

Research Article

Encoding Test Pattern of System-on-Chip (SOC) Using Annular Scan Chain

Guilin Huang, Zhengjin Zhang, Honghai Wang, Jiabao Jiang, and Qilin Wu

School of Information Engineering, Chaohu University, Hefei, Anhui, China

Correspondence should be addressed to Guilin Huang; 747818979@qq.com

Received 14 March 2022; Revised 10 May 2022; Accepted 24 August 2022; Published 2 September 2022

Academic Editor: Jinguang Han

With the improvement of System-on-Chip integration, the chip requires an increasingly large amount of test data. To solve the contradiction between the storage capacity and bandwidth of automatic test equipment (ATE), a new method of test data compression/decompression is proposed based on an annular scan chain. Corresponding fault bits of different test patterns are incompatible, moving test patterns in an annular scan chain, makes all of the new corresponding bits of different test patterns be compatible or backward-compatible, so different adjacent test patterns form a new relation that are indirectly compatible or indirectly backward-compatible, achieves the purpose of test data compression by encoding these indirectly compatible test patterns or indirectly backward-compatible test patterns. According to experimental results, the average compression ratio increases by %6.94 to % 15.1 compared with the other schemes, relative decompression architecture is simple. In the annular scan chain, the test pattern moves clockwise with the minimal bits, generating subsequent test patterns quickly, it is advantageous to reduce the test application time of a single IP core.

1. Introduction

The rapid development of integrated circuit technology has improved the design and manufacturing capabilities of integrated circuits. Microelectronic products that perform composite functions are integrated into a die due to the advancement in IC and semiconductor technology. The improved fabrication technology leads to billions of transistors on an IC with all functionality required for a system which is called system on chip, which continuously brings a series of new challenges to the test [1, 2, 3]. The significance of testing integrated circuits lies in finding out the problems existing in the design or manufacturing process of circuits as early as possible and finding out the defective or faulty chips. Chips are widely used in various electronic devices, such as aviation, medical care, transportation, and household appliances. Whether the chips can work normally relates to the reliable operation of the devices and the safety of people's lives and property, so it is of great significance to ensure the reliable and trouble-free operation of the chips.

Techniques to reduce test cost includes test scheduling based on the system and test data compression based on a single IP core [4, 5]. Increasing test data is a challenge that chip testing must face. Limited ATE storage space and I/O bandwidth store and transmit huge test data, which will improve test cost and lengthen test time, test data compression technology has been trying to solve this test problem [6, 7].

At present, many mature test data compression schemes are proposed, which can be divided into three categories [8, 9]: compression method based on LFSR structure [10], compression method based on broadcast scan [11, 12], encoding compression method [13, 14, 15]. And encoding compression is a very popular method, encodes the data block according to the corresponding relation of the codewords, such as GOLOMB [16], FDR [17], ALT-FDR [18], EFDR [19], 9C [20], BM [21], RL-HC [22], SHC [23], and VIHC [13]. These are all very classic and excellent schemes, the compression effect it can achieve is still very ideal, but it is difficult to accept in the time of test pattern generation. Test data is generated in the unit of the data

block, the whole test set cannot be generated quickly. This paper presents a test pattern encoding method based on the annular scan chain. The proposed scheme moves all test pattern in a clockwise direction, then finds all indirectly compatible or backward-compatible test patterns. Encodes test pattern according to the minimal bits of moving test patterns in the annular scan chain, achieves the purpose of lossless compression.

Contribution of this paper: (1) Compression ratio increases compared with the other schemes. (2) Besides the conventional control Unit, only MUX and inclusive-OR gate are needed by decompression architecture. Decompression architecture is simple. (3) According to the sequence of linear generation graph, the test pattern moves clockwise with the minimal bits generating subsequent test patterns quickly, reduces the time of test pattern generation, helps reduce the test application time of a single IP core.

2. Proposed Test Data Compression Scheme

2.1. Annular Scan Chain. When the test patterns are loaded into the circuit under test, the circuit will give a test response. After the test response is compressed by the test response compressor, it is sent back to the comparator of ATE and compared with the expected test response of the fault-free circuit. If the test responses of both are the same, it indicates that the tested circuit is considered fault-free; otherwise, it is faulty. With the improvement of integrated circuit integration, more and more IP cores are integrated on the circuit, which increases the complexity of the circuit and makes the test of the circuit more and more complicated. Only relying on the special test equipment ATE to test, the test cost is increasing constantly, so the test depends on testability design.

In order to match the design of the test wrapper, a register is added to the function input of the circuit under test, which is called the boundary scanning register. A boundary scan register is also added to the function output of the circuit under test. As shown in Figure 1, there are four boundary scan registers 1, 2, 3, and 4. During the test, registers 1 and 2 are used to store the function input, that is, to store the test bit, and registers 3 and 4 are used to capture the test response, that is, to store the test response. At this time, the scan chain length of the circuit under test is $m\ (x + y + z)$. Boundary scanning is an application of scanning path in the I/O boundary. Scanning design can provide controllability and observability for testing.

The integrated circuit includes application logic, related input and output, and a scanning path composed of BSCs, in which case each pin is connected to a BSC. As shown in Figure 2, BSC structures are interconnected, forming a scanning path between the input end (TDI) and the output end (TDO) of the integrated circuit. During the normal operation of IC, input and output signals pass through the BSC module from NDI to NDO, respectively. When entering the boundary test mode, the test data moves in serial mode from the TDI, the test response is moved out from the TDO serially and observed.

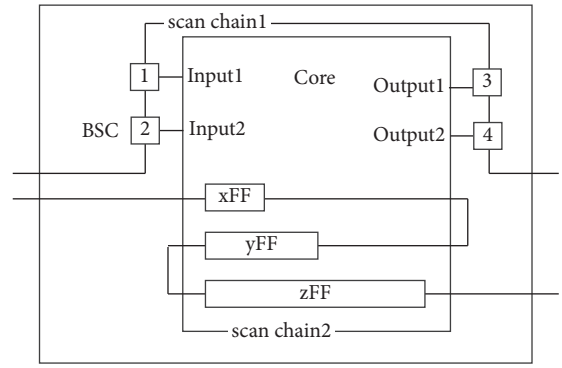

Figure 1: Boundary scanning design.

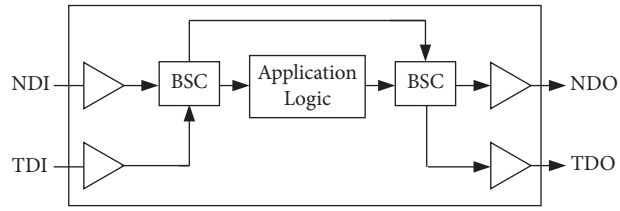

Figure 2: Boundary scan cell structure.

Definition 1. All flip-flops in the chip test wrapped are connected together in a string, forming a single unidirectional scan chain. Input is the functional input terminal (external input) of the sequential circuit, be used to transmit partial data of test patterns. Output is the functional output terminal (external output) of the sequential circuit, be used to transmit partial data of test response. The Q of each flip-flop is connected to the input of the combinational circuit in the sequential circuit, and also to the Q' of the flip-flop unit of the next stage. The first flip-flop in the scan chain is connected with the last flip-flop, that is, the Q of the last flip-flop is also used as the input terminal TD of the first flip-flop, as shown in Figure 3. The dotted circuit structure in the figure connects with the original flip-flops to form an annular scan chain. test patterns can be moved clockwise in the scan chain circularly, therefore, it is called an annular scan chain.

Definition 2. The test pattern moves clockwise in the scan chain, forming a logical annular test pattern. test pattern 1: $A0, A1, A2, \ldots$, An; test pattern 2: $B0, B1, B2, \ldots, Bn$; If $A0$ and $B0$, $A1$ and $B1$, $A2$ and $B2$, $\ldots$, An and Bn are all compatible or backward-compatible, test pattern 1 and test pattern 2 is directly compatible or directly backward-compatible.

Test pattern1 (x0011) moves clockwise one bit in the annular scan chain forming test pattern 1x001. Test pattern 1x001 is compatible with test pattern 2 (1xx01), is backward-compatible with test pattern3 (01x10). Test pattern1 is said to be indirectly compatible with test pattern2, and indirectly backward-compatible with test pattern3.

2.2. Linear Generation Graph. Each test pattern can be regarded as an annular test pattern in the scan chain logically. Firstly, constructs an annular generation graph, nodes

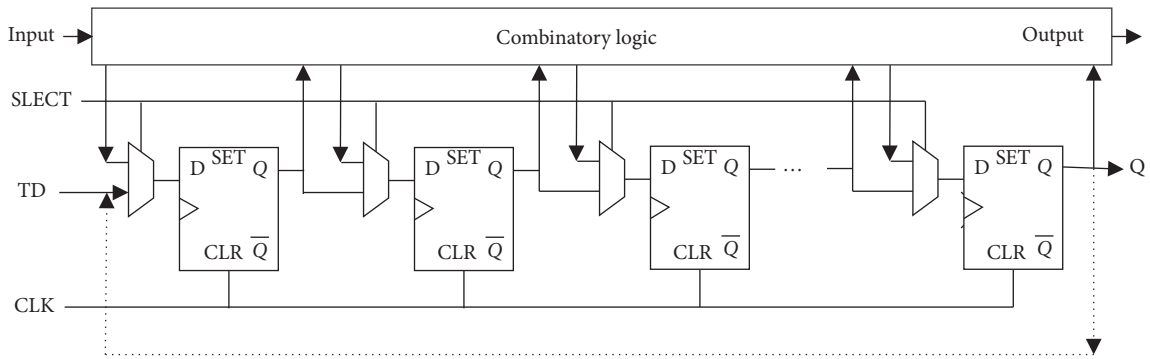

Figure 3: Annular scan chain.

are test patterns, out-degrees, and in-degrees of all nodes are 1, there are N edges in N nodes. Secondly, constructs a linear generation graph, the out-degree of the head node is 1, and the in-degree of the tail node is 1, the out-degree and in-degree of each other node are 1, and there are $N-1$ edges. They are indirectly compatible or backward-compatible between nodes, the bits of moving clockwise is the weight of the graph. Test pattern 0 is expressed as $T(0)$. Bit n of test pattern is expressed as B (n).

Assumes that the test pattern length is L, the number of test patterns is N.

Step 1: the initial state of list List1 is an empty set, and the initial state of list List2 contains the whole test set, that is, all test patterns of the circuit under test.

Step 2: test pattern $T(0)$ enters List1 from List2, then there is no $T(0)$ in List2. In step 3 to step 7, the union of List1 and List2 is the whole test set of the circuit under test, that is, all test patterns of the circuit under test.

Step 3: the length of the test pattern is L, and the test pattern $T(1)$ moves n bits clockwise, then the $B(n)$, $B(n+1)$, ..., $B(L-1)$, $B(0)$, $B(1)$, ...and $B(n-1)$ of the annular test pattern $T(1)$, is corresponding to the $B(0)$, $B(1)$, $B(2)$, ..., $B(L-2)$, B $(L-1)$ of the annular test pattern $T(0)$, respectively, counts the number of incompatible bits between $T(0)$ and $T(1)$ and counts the number of compatible bits between $T(0)$ and $T(1)$. The range of n is between 1 and $L-1$. According to different n, counts the number of incompatible bits and compatible bits between $T(0)$ and $T(1)$ for $L-1$ times. In this statistical process, if the minimal number of incompatible bits is 0, it indicates that $T(0)$ and $T(1)$ are indirectly compatible, then $T(1)$ enters List1 from List2. If the minimal number of compatible bits is 0, it indicates that $T(0)$ and $T(1)$ are indirectly backward-compatible, then $T(1)$ enters List1 from List2.

Step 4: repeats this process, that is, compatibility analysis between any one test pattern $T(i)$ in List1 and certain test pattern $T(j)$ in List2. If $T(i)$ is indirectly compatible or backward-compatible with $T(j)$, then $T(j)$ enters List1 from List2.

Step 5: if any test pattern in List1 is neither indirectly compatible nor backward-compatible with any test pattern in List2, then selects any test pattern $T(k)$ in List1 to move clockwise. When $T(k)$ moves by one bit every time clockwise, it is judged whether it is indirectly compatible or backward-compatible with other test patterns. If it is found that $T(m)$ is indirectly compatible or backward-compatible with $T(k)$ firstly, then $T(m)$ is set as the adjacent node of $T(k)$. $T(m)$ will be removed from List1.

Step 6: repeats step 5 until List1 is an empty set. The last test pattern $T(n)$ becomes the adjacent node of $T(k)$, forming an annular generation graph as shown in Figure 4.

Step 7: removes the edge with the largest weight and construct the directed linear generation graph as shown in Figure 5. In the test process, test patterns will be generated according to the sequence of directed linear generation graph, the seed test pattern $T4$ will generate test pattern $T1$, $T1$ will generate $T2$, and so on until $T6$ is generated, and all test patterns will be generated.

Step 8: selects the first node $T4$ on the linear generation graph as the seed, and other nodes are encoded by the number of bits moved clockwise.

Step 9: returns step 3 to step 8, and so on and constructs a linear generation graph for the rest of the test patterns in List2 until all the test patterns enter the linear generation graph and completes the encoding.

In order to describe the generation process of linear generation graph, provide pseudocode as follow.

Assumes that List2 includes all original test patterns in test set, List1 includes test patterns, which is indirectly compatible or backward-compatible with TV1. TV1 is selected from List2, which is the first element added to the null set List1. TVi stores the result that TV1 moves i bits clockwise in the annular scan chain, TVj represents the test pattern j in test set. TV stores the calculation result that TVi and TVj perform exclusive-or operation by bit. TVk represents the bit k of TV. TVB stores the calculation result that TVi and TVj perform inclusive-or operation by bit. TVBk represents the bit k of TVB.

Assumes that $0 \oplus X == 0$, $1 \oplus X == 0$, $X \oplus X == 0$. Assumes that $0 \odot X == 0$, $1 \odot X == 0$, $X \odot X == 0$.

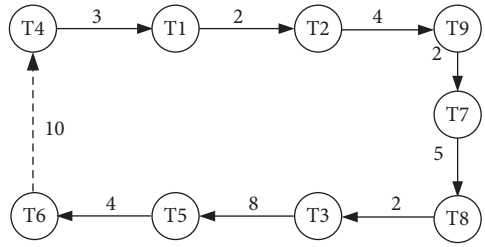

Figure 4: Annular generation graph.

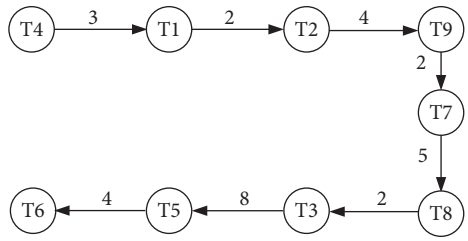

Figure 5: Linear generation graph.

do.
{
$i = 1$;
do.
{$t = 0$;
TVi=((TV1 << L − i)|(TV1 >> i));
do.
{t++;
$j = 2$;
TV = TVi⊕TVj;
TVB = TVi⊙TVj;
if ($\sum_{k=0}^{L-1} TV_k == 0 || \sum_{k=0}^{L-1} TVB_k == 0$).
{
List1.append (TVj);
List3.append (t);
List4.append (j);
$t = 0$;
break;
}
j++;
}while ($j \le$ Num);
i++;
}while ($i \le$ L);
m = len (list3);
list3.append (L−$\sum_{x=1}^{m}$ list3[x]);
y = max {list3 [x] |$0 \le x \le m$};
z = list3.index (y);
selects list3 [$z - 1$] as seed test pattern;
repeat the above calculation.
}while (len (list2) ≠ 0);

2.3. Encoding Modes And Applications. There are 9 test patterns as shown in Table 1, each of which is 40 bits, and all the 9 test patterns are indirectly compatible or backward-compatible. During the test, subsequent test patterns will be generated in the sequence of linear generation graph. $T1$ and its subsequent test patterns move clockwise by one bit every time, find out whether there is a test pattern that is indirectly compatible or backward-compatible with it.

When $T1$ moves clockwise by 2 bits, it is found that the newly formed test pattern is backward-compatible with $T2$, and $T2$ is indirectly backward-compatible with the first test pattern $T1$ at first. Therefore, $T2$ is encoded as100010.

$T2$ continues to move clockwise, when the $T2$ cycle moves clockwise by 4 bits, it is found that the newly formed test pattern is directly compatible with $T9$ at first, and $T9$ is indirectly compatible with $T2$, $T9$ is encoded as 110100.

And so on, $T7$, $T8$, $T3$, $T5$, $T6$, and $T4$ are encoded as 100010, 110101, 110010, 111000, 110100, and 11+ seed, respectively. Selects the first test pattern $T4$ as the seed test pattern, after removing the edge with the largest weight 10.

During the test, according to the sequence of the linear generation graph, $T4$ moves clockwise by 3 bits to get $T1$, and $T1$ generates $T2$, until $T6$ is generated. The sum of moving bts is 31, that is, the sum of weights is 31, which is less than the test pattern length. Generally, the number of clock cycles for test pattern generation in the linear generation graph is far less than the length of a single test pattern.

3. Proposed Decompression Architecture

The decompression architecture of this scheme is mainly composed of a control Unit, MUX, inclusive-OR gate. As shown in Figure 6, en is input enable signal, bit_in is used to transmit compressed test data including the mode, the seed and the encode to the control Unit. Mod1 is the mux channel selection signal. Seed and Q are connected to the two input terminals of MUX, the input terminal Q is connected to the end of scan chain, and the output terminal TD is connected to the head of scan chain. If mod2 is 1, mod2 is used to reverse the Q signal.

(1) The test pattern enters the scan chain. The first input of MUX is gated, TD outputs seed, and the seed test pattern enters the scan chain, that is, bit_in inputs the seed test pattern into the scan chain through the seed channel. After each clock cycle, one bit of the test pattern is moved into the scan chain. After L clock cycles, the L bits of test pattern are moved in, that is, the whole seed test pattern is moved into the scan chain.

(2) According to the test pattern encode, moving generates the next new test pattern. The second input of MUX is gated, TD outputs Q, and the original test pattern in the scan chain moves clockwise to generate a new next test pattern. In the process of moving clockwise, in the first clock cycle, the data in the flip-flop L enters the first flip-flop of the scan chain. Similarly, the data in flip-flop 1 enters flip-flop 2, the data in flip-flop 2 enters flip-flop 3, ..., and the data in flip-flop L-1 enters flip-flop L. After one clock cycle, all the test data moves one bit clockwise in the scan chain. Test data moves clockwise as many bits as the clock cycles.

Table 1: Compatibility analysis among test patterns.

Number	Test pattern	Bits	Mode1	Mode2	Encode
*T*4	01x1001101101000110101010101001111001010	3	0	—	Seed
*T*1	0100x1100110110100011010101010011111001	2	1	1	0011
*T*2	10101100011x0100101110010101010101100001	4	1	0	0010
*T*9	000110101100011x0100101110010101010101110	2	1	1	0100
*T*7	01111001010011100x1011010001101010101010	5	1	0	0010
*T*8	01010011x1001010011100110110100011010101	2	1	1	0101
*T*3	010101001x1100101001110011011010000110101	8	1	1	0010
*T*5	001101010101x1001111001010011100110110 10	4	1	1	1000
*T*6	10100011010101x1010011110010100111001101	(10)	1	1	0100

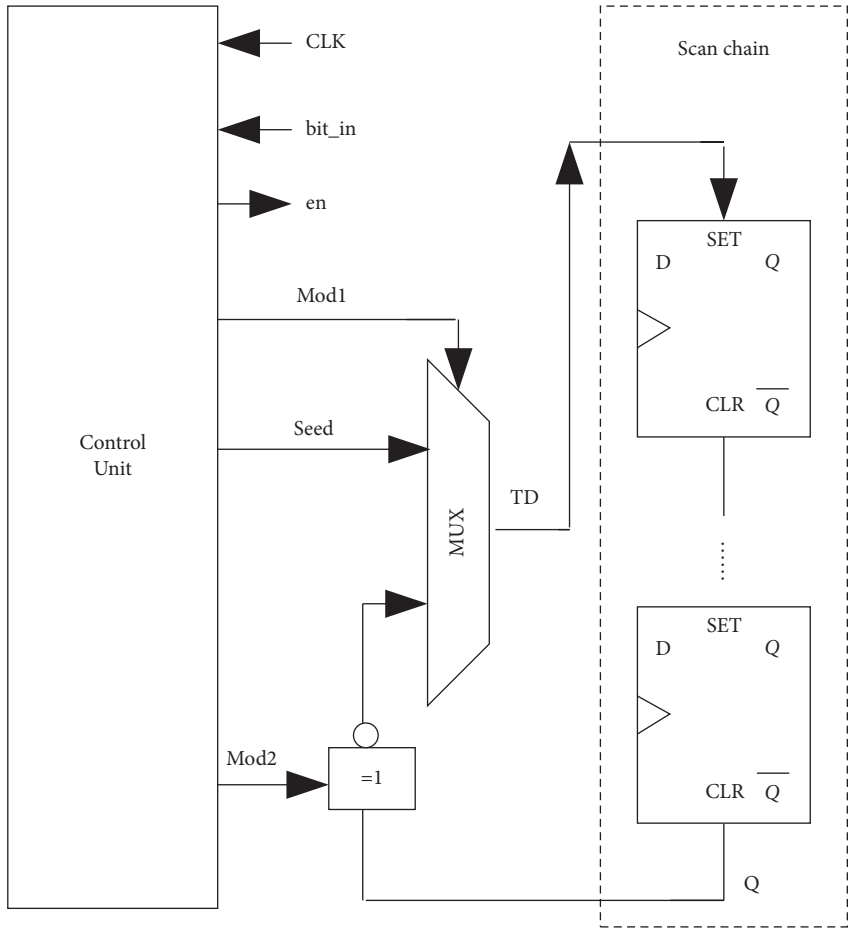

Figure 6: Decompression architecture.

(3) Repeat (2), continuously generates the next test pattern until all the test patterns in the linear generation graph are generated according to to encode.

4. Experiments

4.1. Compression Analysis. Time complexity of constructing an annular generation graph is O (N^*L), and the time complexity of constructing a linear generation graph is O (L), then the time complexity of the whole algorithm is O (L), and the algorithm is simple.

Obviously, the sum of weights D $[i][i+1]$ in annular generation graph is L, and the sum of weights in the linear generation graph is less than L. Except for the seed test pattern, the number of clock cycles for test pattern generation in the linear generation graph is Nu.

$$\begin{aligned} Nu &< \sum_{i=1}^{N} D[i][i+1] + \left(\log_2 L + 2\right) * N \\ Nu &< L + \left(\log_2 L + 2\right) * N \\ \frac{nu}{N} &< 1 + \left(\log_2 L + 2\right). \end{aligned} \tag{1}$$

For the experimental circuit, the values range of $\log_2 L$ is [5, 11], In formula (1), N is greater than 2, with the increase

Table 2: Experimental results.

Experimental circuit	FDR	ALT-FDR	EFDR	9C	BM	RL-HC	SHC	VIHC	GOLOMB	Proposed scheme
s5378	47.98	50.77	53.67	51.64	54.98	53.75	55.10	51.52	37.11	74.3
s9234	43.61	44.96	48.66	50.91	51.19	47.59	54.20	54.84	45.25	59.6
s13207	81.30	80.23	82.49	82.31	84.89	82.51	77.00	83.21	79.74	88.47
s15850	66.21	65.83	68.66	66.38	69.49	67.34	66.00	60.68	62.82	73.94
s38417	43.37	60.55	62.02	60.63	59.39	64.17	59.00	54.51	28.37	63.1
s38584	60.93	61.13	64.28	65.53	66.86	62.40	64.10	56.97	57.17	69.72
Average	60.01	60.58	63.55	63.35	65.48	62.72	63.28	61.44	56.42	71.52

of N, the range of average time nu decreases linearly, which is far less than 1, and the average number of test clock cycles of each test pattern is far less than the length of the test pattern. The compression ratio is calculated by $(T_D - T_E)/T_D \times 100$. $\mu = T_D - T_E$.

$$\mu = N * (L - 2 - \log_2 L) + \log_2 L + 1 - L. \quad (2)$$

The variation range of $\log_2 L$ is very small. Therefore, it can be seen from formula (2) that the compression ratio is determined by N, that is, the number of test patterns in the linear generation graph. Therefore, this paper is committed to solving the problem that makes more test patterns enter the linear generation graph.

$$\frac{d\mu}{dN} = L - 2 - \log_2 L. \quad (3)$$

Here, $d\mu/dN > 0$, it means that when L is constant and N increases successively, the compression ratio is improving constantly.

4.2. Experiment Comparison. Selected 6 large scale circuits from ISCAS′89 benchmark circuits as the experimental circuit as shown in Table 2, used atalanta to generate original test patterns which is will be encoded.

In order to prove the simplexity of decompression architecture, used the Design Compiler of Synopsys Company to synthesize and simulate, and calculated the area of the decompression architecture. The hardware area overhead of decompression architecture is 4684.32 μm^2.

In order to evaluate the performance of this scheme in compression, compressed test patterns of ISCAS′89 benchmark circuits using Python. During the experiment, test patterns are continuously injected into the compression program module, during the test pattern injection process, the compression ratio keeps upward trends as shown in Figure 7. We can see that the test data compression scheme proposed in this paper is robust and has practical application value, meets the expectation of formula (1). The abscissa indicates the number of test patterns injected into the algorithm. The ordinate indicates the compression ratio.

The last column of Table 2 shows the compression ratio of this scheme. The compression ratio is better than other schemes, which proves that the scheme in this paper has good adaptability, and the maximum compression ratio reaches 88.47%.

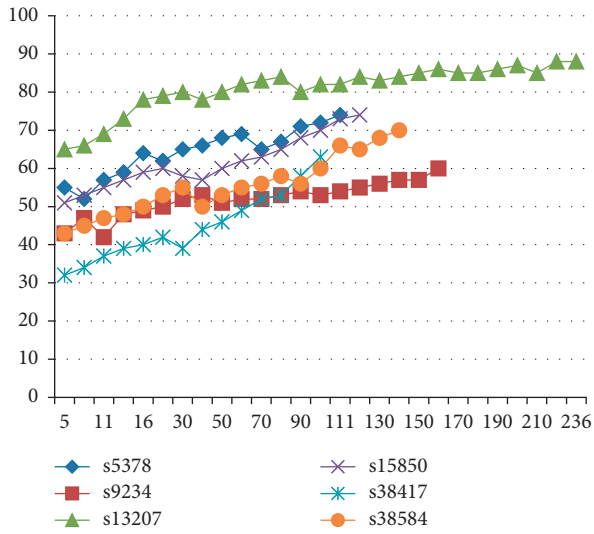

Figure 7: Compression ratio change curve.

5. Conclusions

In order to cover all faults of the test circuit, the corresponding bits of different test patterns are incompatible. This paper mainly explores the compatibility between different test patterns. Moves the test patterns in the scan chain clockwise, staggers the corresponding bits, and corresponding bits are no longer corresponding.

There are a large number of irrelevant bits in the test patterns, newly formed corresponding bits in test patterns are all compatible or backward-compatible after moving clockwise, eliminates the incompatibility between the original corresponding bits, solves the incompatibility between different test patterns, makes different test patterns have a new relationship which is indirectly compatible or backward-compatible, and provides a basis for test data compression.

Every time the seed test pattern moves 1 bit clockwise, find out whether there is a test pattern that is indirectly compatible or backward-compatible with it, test patterns that are indirectly compatible or backward-compatible with it will be constructed into the linear generation graph. According to the sequence of directed linear generation graph, the test pattern moves clockwise with the smallest amplitude, and the subsequent test patterns are generated quickly, reduces the time of test patterns generation and reduces the test application time effectively.

Data Availability

We evaluate the performance of the proposed scheme based on the ISCAS′89 benchmark circuits, and our model and related hyperparameters are provided in our paper.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by Key Projects of Natural Science Research in Universities of Anhui Province: Research and Implementation of Decoding Unit for Ternary Optical Processor (Grant nos. KJ2020A0681 and KJ2019A0682), The Provincial Natural Science Research Program of Higher Education Institutions of Anhui Province (Grant no. KJ2021A1030), The Quality Improvement Project of Chaohu university on Discipline Construction (Grant no. kj21gczx03), Special Support Plan for Innovation and Entrepreneurship Leaders in Anhui Province, and Key Discipline of Computer Science and Technology of Chaohu University (Grant no. kj22zdxk01).

References

[1] G. Chandrasekaran, S. Periyasamy, and K. P. Rajamanickam, "Minimization of test time in system on chip using artificial intelligence-based test scheduling techniques," *Neural Computing & Applications*, vol. 32, pp. 5303–5312, 2020.

[2] G. Chandrasekaran, "Test scheduling of system-on-chip using dragonfly and ant lion optimization algorithms," *Journal of Intelligent and Fuzzy Systems*, vol. 40, no. 3, pp. 4905–4917, 2021.

[3] G. Chandrasekaran, S. Periyasamy, and P. R. Karthikeyan, "Test scheduling for system on chip using modified firefly and modified ABC algorithms," *SN Applied Sciences*, vol. 1, no. 9, p. 1079, 2019.

[4] S. Huhn, D. Tille, and R. Drechsler, "Hybrid architecture for embedded test compression to process rejected test patterns," in *Proceedings of the European Test Symposi-um (ETS) 2019 IEEE European*, pp. 1-2, Baden Baden, Germany, May, 2019.

[5] S. Rooban and R. Manimegalai, "Prediction of theoretical limit for test data compression," in *Proceedings of the International Conference on Recent Trends in Advance Computing (ICRTA-C)*, pp. 41–46, Chennai, India, November, 2018.

[6] G. Chandrasekaran, V. Kumarasamy, and G. Chinraj, "Test scheduling of core based system-on-chip using modified ant colony optimization," *Journal Européen des Systèmes Automatisés*, vol. 52, no. 6, pp. 599–605, 2019.

[7] G. Chandrasekaran, G. Singaram, R. Duraisamy, A. S. Ghodake, and P. K. Ganesan, "Test scheduling and test time reduction for SoC by using enhanced firefly algorithm," *Revue d'Intelligence Artificielle*, vol. 35, no. 3, pp. 265–271, 2021.

[8] H. Yuan, K. Guo, X. Sun, J. Mei, and H. Song, "A power efficient BIST TPG method on don't care bit based 2-D adjusting and hamming distance based 2-D reordering," *Journal of Electronic Testing*, vol. 31, no. 1, pp. 43–52, 2015.

[9] H. Vohra and A. Singh, "Test data compression using hierarchical block merging technique," *IET Computers & Digital Techniques*, vol. 12, no. 4, pp. 176–185, 2018.

[10] M. Yi, H. Liang, and K. Zhan, "Optimal LFSR-coding test data compression based on test cube dividing," in *Proceedings of the International Conference on Computational Science and Engineering (CSE 2009)*, pp. 698–702, Vancouver, Canada, August, 2009.

[11] S. Mitra and K. Kim, "XPAND: an efficient test stimulus compression technique," *IEEE Transactions on Computers*, vol. 55, no. 2, pp. 163–173, 2006.

[12] Y. Yu, X. Peng, and Y. Zhang, "Test data compression method for multiple scan chains based on repeated subvectors," *Chinese Journal of Scientific Instrument*, vol. 30, no. 2, pp. 356–361, 2009.

[13] M. Zhang, Z. Xia, and J. Kuang, "Improving compression ratios for code-based compressions by test set significant components based transform-decomposing method," *Journal of Electronic Measurement and Instrument*, vol. 34, no. 09, pp. 94–100, 2020.

[14] H. Wu, W. Zhan, and Y. Cheng, "Dictionary encoding method based on independent of test data," *Journal of Electronic Measurement and Instrument*, vol. 30, no. 4, pp. 638–644, 2016.

[15] W. Zhan and S. Zhao, "Compatible compression method for a maximum of approximately compatible grouping test patterns," *Journal of Electronic Measurement and Instrument*, vol. 30, no. 12, pp. 1933–1940, 2016.

[16] A. Chandra and K. Chakrabarty, "System-on-a-chip test-data compression and decompression architectures based on Golomb codes," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 3, pp. 355–368, 2001.

[17] 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 Transactions on Computers*, vol. 52, no. 8, pp. 1076–1088, 2003.

[18] J. Kuang, Y. Zhou, and S. Cai, "Adaptive EFDR coding method for test data compression," *Journal of Electronics and Information Technology*, vol. 37, no. 10, pp. 2529–2535, 2015.

[19] W. Zhan, Y. Cheng, H. Wu, and J. Jiang, "Automatical generation method of generalized folding set by guiding test patterns," *Journal of Southeast University (Natural Science Edition)*, vol. 48, no. 02, pp. 265–269, 2018.

[20] M. Tehranipoor, M. Nourani, and K. Chakrabarty, "Nine-coded compression technique for testing embedded cores in SoCs," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 13, no. 6, pp. 719–731, 2005.

[21] A. H. El-maleh, "Efficient test compression technique based on block merging," *IET Computers & Digital Techniques*, vol. 2, no. 5, pp. 327–335, 2008.

[22] M. Nourani and M. H. Tehranipour, "RL-huffman encoding for test compression and power reduction in scan applications," *ACM Transactions on Design Automation of Electronic Systems*, vol. 10, no. 1, pp. 91–115, 2005.

[23] A. Jas, J. Ghosh-dastidar, and N. G. Mom-eng, "An efficient test vector compression scheme using selective Huffman coding," *IEEE Transaction on. Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 6, pp. 797–806, 2003.