Test Pattern Compression Using Prelude Vectors In Fan-out Scan Chain with Feedback Architecture

Nahmsuk Oh, Rohit Kapur, T. W. Williams, and Jim Sproch
Synopsys Inc., 700 E. Middlefield Road, Mountain View, CA 94043
{nahmsuk, rkapur, tww, jims}@synopsys.com

Abstract

This paper proposes a new test compression technique that employs *Fan-out SCAN chain with Feedback* (FSCANF) architecture. It allows us to use *prelude vectors* to resolve dependencies created by fanning out multiple scan chains from a single scan-in pin. This paper describes the new proposed architecture as well as the algorithm that generates compressed test vectors using vertex coloring algorithm. The distribution of specified bits in each test pattern determines the compression ratio of the individual test pattern. Therefore, our technique optimizes the overall compression ratio and shows higher reduction in test data and application time than previous techniques, which use the extreme case of serializing all the scan chains in the presence of conflicts across the fan-out scan chains. The FSCANF architecture has small hardware overhead and is independent of scan cell orders in the scan chains. Experimental results show that our technique significantly reduces both the test data volume and test application time in six of the largest ISCAS 89 sequential benchmark circuits compared to the previous techniques.

1. Introduction

DFT technology has matured over the years with Scan Design becoming common to most design flows. The test industry is now focused on optimizing the scan design techniques to minimize the cost of test as measured by the test data volume and more importantly the test application time. The time taken to apply a scan test pattern is dominated by the shift operation of the scan chain. To reduce this time, the general approach taken is to create more parallel scan chains. Since the number of flip-flops in the design is a constant, more parallel scan chains means fewer flip-flops per scan chain, resulting in shorter chains and thus shorter application time. In the traditional approaches, however, the number of available I/O pins on the chip limits the number of parallel scan chains.

One notable approach is that of encoding a test pattern into a seed, which is applied through fewer inputs and the decoding process on chip (possibly LFSR based), could transmit the data to many more scan chains than the physically supported though the I/O's of the chip. It allows us to reduce not only test application time but also test data volume. Figure 1.1(a) is a pictorial representation of this concept. This approach relies on considerable amount of DFT overhead to solve the problem.

Another approach is the *fan-out scan chain architecture*, in which a single scan input fans out to multiple chains (Figure 1.1 (b)). This approach creates significant dependencies in values across scan chains because the same vector is shifted into those scan chains. For example, the benefits of this approach come when the dependencies in the values of the scan chains do not interfere with the requirements of the test patterns. However, when conflicts are encountered across scan chains, a very conservative configuration is so far adopted to provide the problematic test patterns - All the scan chains are serialized. This approach has significant benefits because of its simplicity in DFT. However, the benefits are dependent on the ability to create test patterns that do not require conflicting values from scan chains that are dependent on each other because of the fan-out of the scan inputs.

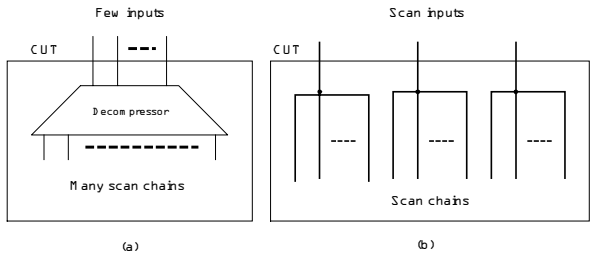

Figure 1.1. (a) Decompression based approaches for short parallel scan chains. (b)Fan-out scan chain structure to create shorter scan chains with a limited number of scan inputs.

In this paper, we improve on the compression technique that uses fan-out scan chains and create a solution in which dependencies caused by the fan-out scan chain structure do not interfere in the application of test patterns. Conflicts across the fan-out scan chains are handled by using *prelude vectors* whose purpose is not to detect faults but to resolve conflicts in a test pattern. The number of prelude vectors is a function of the number of conflicts in an individual test pattern: fewer prelude vectors for fewer conflicts, and more prelude vectors for more conflicts. Therefore, we avoid the extreme solution of serializing all the scan chains to resolve conflicts.

2. Previous Works

A technique using a single input supporting multiple scan chains was proposed in [1]. The number of scan chains is equal to the number of independent circuits, but this technique requires multiple scan chains to be in independent circuits.

The Illinois Scan Architecture (ILS) shown in Figure 2.1 [2] overcomes this limitation by using two modes of operation. The first mode is called the *broadcast scan mode* in which the same vector is shifted into multiple scan chains through one scan-in pin. This is similar to the fan-out scan chain structure shown in Figure 1.1 (b). The

1530-1591/03 $17.00 © 2003 IEEE

second mode is called the *serial scan mode*, in which all the scan chains are connected in series. The serial scan mode is used whenever the broadcast scan mode cannot be used due to conflicts across multiple scan chains. Since the serial scan mode requires a full scan test pattern, it is very important to reduce the number of test patterns applied in the serial scan mode. In other words, reducing the number of conflicts across multiple scan chains is a key factor for test compression, and heavily dependent on the scan chain configuration, i.e., the number of the scan chains and the assignment of scan cells to the scan chains. Finding the optimum scan chain configuration that will produce the minimum number of test cycles is an NP-hard problem [2]. The ILS uses a heuristic algorithm to achieve optimal mapping of scan cells to scan chains, but requires scan chain reordering. Reordering scan chains is very expensive because it requires changing the place and route information of the scan cells and may change the timing of the circuit [3]. Reconfiguring ILS with a different scan chain length was proposed in [4] to avoid reordering scan chains and reduced the number of undetectable faults in the broadcast scan mode, but yet didn't solve the fundamental problem of breaking dependencies across multiple scan chains in the fan-out scan chain architecture.

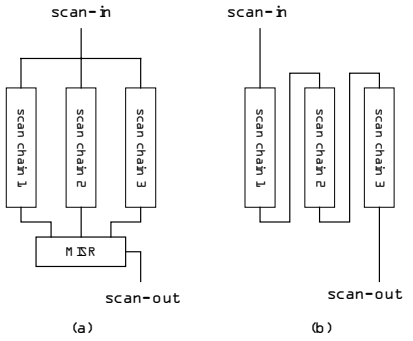

Figure 2.1. Illinois scan architecture: (a) broadcast scan mode, (b) serial scan mode.

3. Fan-out Scan Chain with Feedback (FSCANF) Architecture

3.1. Basic Fan-out of Scan Chains

Let us describe the basic concept of the technique with a small design as an example. The design consists of three scan chains that have three scan cells in each as shown in Figure 3.1. The input pins of the scan chains are tied to one input pin, which is a *scan-in* pin. A *scan-in vector* is a vector that is shifted into the scan-in pin to fill in the scan cells with the required logic values to detect faults. A *test cube* is a two dimensional matrix representation of a test pattern in which each element of the test cube represents a scan-in logic value of one scan cell of the design. In this paper, each column of the test cube represents one scan chain, and the *n*th row of the test cube represents the *n*th scan cells from the input pins of the scan chains. In this example, a test cube is a 3 by 3 matrix.

Suppose three of the nine scan cells are specified bits and the rest of the scan cells are don't cares as shown in Figure 3.1 (a). A test cube for this pattern is shown in Figure 3.1 (b). Since each row of the test cube has only one specified bit, a scan-in vector 011 can fill in the specified bits successfully with the correct logic values as illustrated in Figure 3.1 (c) and Figure 3.1 (d). The rest of the bits are don't cares and filled with the same values as the specified bits in the same rows.

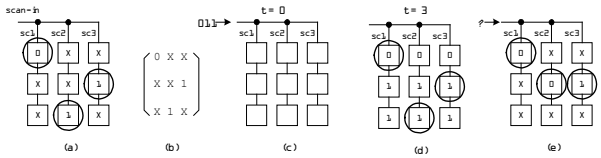

Figure 3.1. A 3 by 3 test cube as an example.

However, consider the test cube in Figure 3.1 (e). Two different logic values are specified in the second row of the cube, i.e., there exists a *conflict* across two scan chains in the second row, and no vector shifted into the scan-in pin can specify the required values. We call this row an *irregular row* of the test cube, because different logic values, 0 and 1, exist in the same row.

3.2. Breaking Dependencies with Prelude Vectors

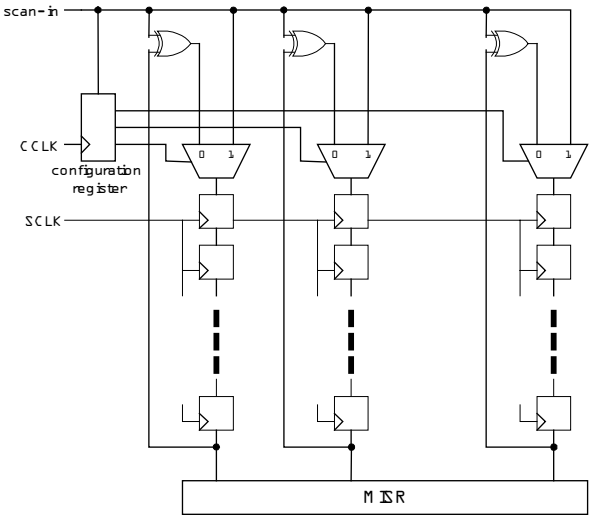

Figure 3.2. The FSCANF architecture.

The FSCANF architecture shown in Figure 3.2 allows for the scan chains to be loaded through one of the two input paths: *loading path* and *feedback path*.

- The loading path supplies values directly from the scan-in pin to the scan chains connected.
- The feedback path supplies values through a XOR filter that allows for the values to be passed through or inverted depending on the pre-existing values in the scan cells of the associated chain.

Furthermore, the FSCAF architecture allows an individual scan chain to select filtered values or direct values through a configuration register. The *i*th bit of the configuration register is connected to the multiplexor of the *i*th scan chain and selects the input path of the scan

chain. The configuration register is loaded by separate clocking signal CCLK. It might have a reset pin for fast restoration of broadcast mode.

The filtering mechanism is dependent on pre-loaded values in the scan chains. These values are loaded as a *prelude vector* into the scan chains. The values in the prelude vector perform XOR operation on the next scan-in vector in selected scan chains to load conflicting values in irregular rows. Thus, the prelude vector is the vector whose purpose is not to detect faults but to load certain values onto the scan chains so that the next scan-in vector can specify bits in irregular rows.

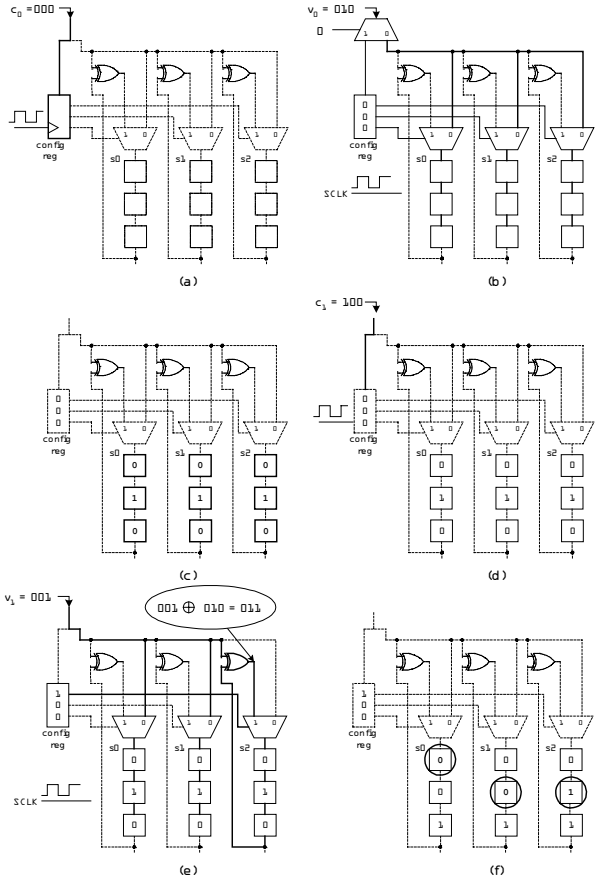

Figure 3.3. Conflict resolution example.

Let us illustrate how this mechanism works by taking the same example shown in Figure 3.1 (e). In the first stage, we load a configuration vector c_0 = 000 to the configuration register by clocking CCLK as shown in Figure 3.3 (a). In this paper, we assume that the rightmost bit of the vector is first shifted into the scan chain and the leftmost bit last shifted into the scan chain. It enables us to load the prelude vector v_0 = 010 into the scan cells by clocking SCLK as shown in Figure 3.3 (b). After v_0 is loaded into the scan cells, the scan cells are filled with the same logic values as the specified bits. A snap shot of the scan cells after loading v_0 is shown in Figure 3.3 (c). In the second stage, we clock CCLK again and load configuration vector c_1 = 100 into the configuration register. In Figure 3.3 (d), the bit on the top of the configuration register is 1 (the leftmost bit of c_1), which configures scan chain s_2 with the feedback path. The scan chain s_0 and s_1 are configured with the loading paths. Therefore, in s_2, an XOR operation is performed on the scan-in vector v_1 with the previously loaded prelude vector v_0, and the new values are shifted into s_2 as shown in Figure 3.3 (e). Since the second bit of the prelude vector v_0 = 010 is 1, the second bit of the scan-in vector v_1 = 001 in s_2 is inverted to 1, which successfully specifies the required logic value in the second scan cell of s_2. Thus, we can specify different logic values in the irregular row and fill in the scan cells with the required specified values as shown in Figure 3.3 (f).

In our technique, a compressed test vector of a test pattern is one of the following: (1) a scan-in vector (2) a configuration vector and a prelude vector (3) a configuration vector and a scan-in vector. In this example, two test vectors, (v_0 c_0) = 010 000 and (v_1 c_1) = 001 100, were obtained, and the total number of bits is 12. Since the number of scan cells is 9, we did not reduce the size of the test pattern in this small example. However, the reduction ratio would be higher if the scan chain contains more cells in it. For example, if each of three scan chains contains 100 cells, the size of the test vector is 103 (3 bits for the configuration vector and 100 bits for the scan-in vector). There are 300 cells in the CUT, thus, the reduction is 31% with a prelude vector (1 – 2*103/300). Furthermore, if no irregular row exists in the test cube, the reduction would be 66% (1 – 103/300). This illustrates how the compression ratio changes from one pattern to another based on the distribution of specified bits.

3.3. Test Vector Computation Algorithm

In general, multiple prelude vectors are required to specify different values in the multiple irregular rows of a test cube. The prelude vector computation algorithm is shown in Figure 3.4. The *conflict graph* described in the algorithm is the graph where the vertices represent the scan chains, and the edges between two vertices shows that two scan chains have different logic values in the same irregular row. A minimum vertex coloring of that graph yields the minimum number of prelude vectors that are needed.

```
1  /* construct a conflict graph */
2  Identify irregular rows in the test cube
3  For each irregular row
4     If scan chain sk and sl have different specified bits,
5           Add an edge (sk, sl) to the conflict graph
6
7  /* apply the vertex coloring algorithm */
8   Apply the vertex coloring algorithm to the graph and number the
      colors α0, α1, …, αm
9
10 For each color αi from α0 to αm-1 {
11    /* construct vi */
12    For each scan chain sj with color αi {
13          Specify bits in scan-in vector vi with the specified bits in sj
14    }
15    /* modify vi-1 */
16    vi-1 = vi-1 ⊕ vi,  if i ≠ 0
17    /* construct configuration vector ci */
18    For each scan chain sj with color from α0 to αi-1 if i ≠ 0  {
19          Put 1 in the bit indicating sj in configuration vector ci
```

```
20    }
21 }
```

Figure 3.4. The prelude vector and scan-in vector computation algorithm.

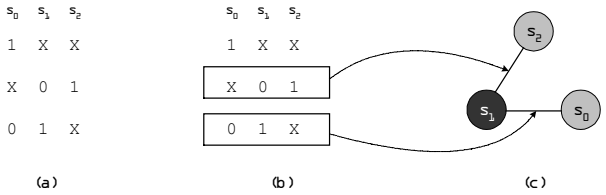

Figure 3.5. (a) An example test cube that has multiple irregular rows and illustrates constructing a conflict graph.

Let us explain the algorithm with the same example previously shown in Figure 3.1, but the test cube has one more specified bit in the third row as illustrated in Figure 3.5 (a). Figure 3.5 (b) shows the two irregular rows that are boxed in the test cube (line 2 of the algorithm). The irregular row in the second row has two different logic values between s_1 and s_2. Thus an edge connecting vertex s_1 and s_2 is drawn in the conflict graph as illustrated in Figure 3.5 (c) (line 3 – 5 of the algorithm). Similarly, the irregular row in the last row draws an edge connecting s_0 and s_1 in the graph. The vertex coloring algorithm colors the graph two colors: s_0 and s_2 with color α_0 (gray) and s_1 with α_1 (black).

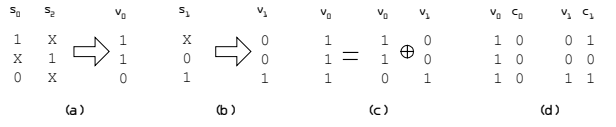

Figure 3.6. (a) The scan-in vector v_0 is obtained from s_0 and s_2 of the test cube shown in Figure 3.1 (a). (b) v_1 is obtained from s_1. (c) v_0 is recomputed by performing XOR operation on it with v_1. (d) The final two test vectors are obtained: (v_0 c_0) = 111 000, (v_1 c_1) = 001 101.

Now, in the lines 10-14 of the algorithm, we pick color α_0 (i = 0) and construct scan-in vector v_0 from the specified bits in s_0 and s_2 that have the same color α_0 (Figure 3.6 (a)). Since i = 0, the lines 15 - 20 are skipped. In the next step, we pick color α_1 (i = 1) and build scan-in vector v_1 as shown in Figure 3.6 (b). The previous vector v_0 is recomputed in line 16. The resulting v_0 is shown in Figure 3.6 (c). Since s_0 and s_2 were colored α_0 (j = 0 and 2), in lines 17 - 20 of the algorithm, we construct a configuration vector c_1 in such a way that s_0 and s_2 are with the feedback path. Finally, Figure 3.6 (d) shows the resulting two test vectors obtained.

Figure 3.7 illustrates the state of the scan cells after loading the prelude vector v_0 and the scan-in vector v_1 shown in Figure 3.6 (d). The required specified bits are shown again in Figure 3.7 (a). Since the configuration vector c_0 is 000, the scan-in vector v_0, 111, is loaded onto the scan chains with the loading paths, and the final values are shown in Figure 3.7 (b). Then, when we load v_1, 001, an XOR operation is performed on v_1 with the previous values (111) in s_0 and s_2 because the configuration vector c_1 = 101. The final values after loading v_1 in the scan cells are shown in Figure 3.7 (c), and we can observe that the required bits are all specified correctly with the desired values.

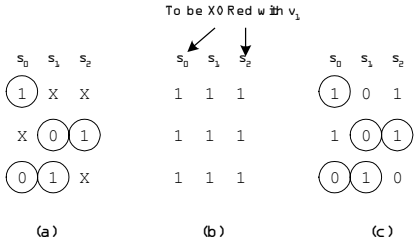

Figure 3.7. (a) The test cube with the specified bits circled. (b) A snap shot of the scan cells after loading the prelude vector v_0. (c) A snap shot after loading the scan-in vector v_1. All the specified bits have required logic values.

As shown in this section, the distribution of the specified bits in test cubes determines the number of prelude vectors. If the number of specified bits is small in the test patterns, the compression ratio will be high because the probability of having irregular rows is very low, and few prelude vectors are needed. On the other hand, if the number of the specified bits is large, the compression ratio will be low because the probability of having one of more irregular rows is very high, thus the number of prelude vectors will increase.

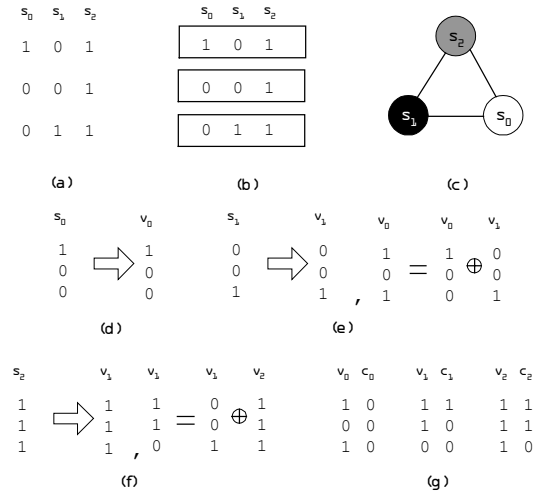

Figure 3.8. (a) A test cube with every bit specified. (b) Three irregular rows boxed. (c) A corresponding conflict graph. The minimum number of colors is three (white, gray, and black). Therefore, two prelude vectors and one scan-in vector are required. (d) v_0 is obtained from s_0. (e) v_1 is obtained from s_1, and v_0 is recomputed. (f) v_2 is obtained, and v_1 is recomputed. (g) The final three test vectors: prelude vector v_0 and v_1, and scan-in vector v_2, and three associated configuration vectors.

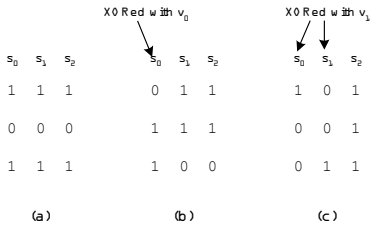

Figure 3.9. (a) A snap shot of the scan cells after loading v_0. (b) A snap shot after loading v_1. (c) A snap shot after loading v_2.

The worst case scenario is to specify every bit in a test cube. Our technique achieves this by having multiple prelude vectors. An example is shown in Figure 3.8, in which each of the steps of the algorithm is shown. We can specify all the bits in the test cube by using two prelude vectors. Snap shots of scan cells at each state after loading prelude vectors are shown Figure 3.9.

3.4. Reducing Test Application Time by Loading Configuration in Parallel

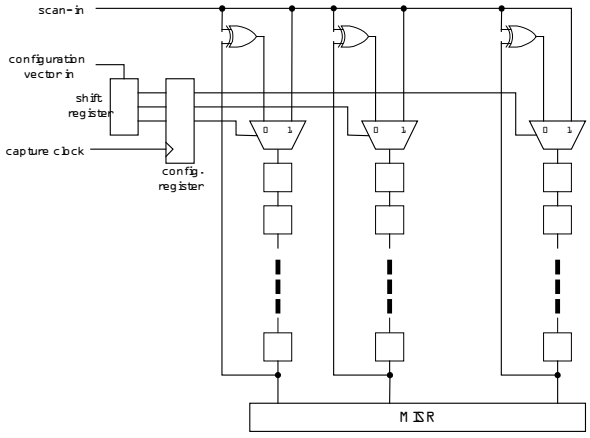

Figure 3.10. FSCANF architecture for parallel configuration load.

A configuration vector for a next vector can be loaded in parallel with a prelude vector or a scan-in vector by adding one more shift registers next to the configuration register as shown Figure 3.10. When a capture clock is applied to all the scan cells, it is also applied to the configuration register, which transfers the value in the shifter register to the configuration register. Therefore, the new configuration is set up for the next vector to be shifted in, and test application time to load configuration vectors can be saved.

4. Experimental Results

In this section, we take the largest of the ISCAS 89 sequential benchmark circuits, and demonstrate how much test data and test application time can be reduced. Table 4.1 shows the characteristics of the six benchmark circuits used in our experiment. The number of scan cells is the number of sequential elements in the circuit. It also shows the average percentage of the specified bits in test patterns, which were obtained by one of the commercial ATPG tools with the highest pattern merge effort. With this highest merge effort option, the ATPG tool repeatedly compresses the test patterns, which often reduces 90% of the original patterns. The number of patterns and the percentage of specified bits shown in this table are obtained after this ATPG tool compression.

Table 4.1. Benchmark circuit characteristics.

Circuit name	s13207	s15850	s38417	s38584	s5378	s9234
Number of scan cells	638	534	1636	1426	179	211
Number of patterns	117	107	417	125	113	128
Specified bits	13.1%	20.1%	6.3%	18.2%	25.2%	26.3%

We show two sets of experimental data: test data volume reduction and test application time reduction. In the test data volume reduction, we obtain the total test data including configuration vectors and compare it with the original test data. In test application time reduction, we obtain the total elapsed time to test the circuit that employs the FSCAF architecture with parallel configuration loading, and compare it with original test time. In contrast to the test data volume reduction, the test application time includes only the elapsed time applying test vectors since configuration vectors can be applied to CUT in parallel with test vectors. The equations to compute the compression ratio for our technique are:

$$\text{Test data volume reduction}: \quad (1 - \frac{\text{scan - in vectors} + \text{prelude vectors} + \text{configuration vectors}}{\text{original test data}}) \times 100\%$$

$$\text{Test application time reduction}: \quad (1 - \frac{\text{test time of applying (scan - in vectors} + \text{prelude vectors)}}{\text{test time of applying original test patterns}}) \times 100\% .$$

For comparison, we also compute test data and application time reduction of ILS, in which one full scan vector is fed into scan chains in the serial scan mode whenever there is one or more irregular rows in the test patterns. Because we assume that the scan chains already exist in the CUT and scan chain reordering is prohibited, the data shown for ILS do not use the optimal mapping method [2] that maps flip flops to scan chains in such a way that the number of serialized patterns is minimized.

The test data volume reduction depends on the number of scan chains. As the number of scan chains increases, the scan chain length decreases resulting in reducing the test data volume. However, the probability of irregular rows increases because each row of the test cube has more scan cells. Thus, the number of prelude vectors as well as the size of the configuration vector increases. For example, as shown in Figure 4.1, the test data volume reduction percentage increases as the number of scan chains increases until it reaches a peak where the increased number of prelude vectors and configuration vectors offsets the reduction of the volume of scan-in vectors. If the number of scan chains keeps growing, the test data volume reduction percentage starts decreasing. On the other hand, fewer scan chains show better results in ILS because the increased number of irregular rows increases the number of full scan vectors in broadcast mode which are required to feed all scan cells in series.

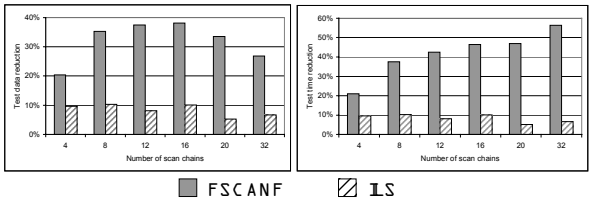

Figure 4.1. Test data and application time reduction of s38584 in FSCANF and ILS with different number of fan-out scan chains. The FSCANF shows a bell-shape curve with the peak at 16 scan chains in test data volume reduction and a continuously increasing curve in test application time reduction.

The test application time reduction does not depend on the size of configuration vectors, and the reduction percentage increases as the number of scan chains grows in our technique. Therefore, the test time is reduced the most with the largest number of scan chains. On the other hand, the previous technique still suffers as the number of scan chains increases because, anyway, the full scan vector is fed into the scan chains in series in the broadcast mode. Thus, our technique shows much better performance in terms of reducing test time.

Changing the number of scan chains in the circuits, we obtained the test data and test application time reduction percentage of our technique and previous technique, and show the results in Table 4.2 and Table 4.3.

The percentage of specified bits in test patterns also plays an important role in reducing the test data volume as well as test time. Our technique shows better results than the previous technique as the number of specified bits increases. For example, specified bits of test patterns in s38417 are only 6.7%. Our technique reduces 74.8% of test data, but the previous technique also reduces 64.5%, which shows only 10% difference. However, consider s5378 whose specified bits in test patterns are 25.2%. The highest reduction is 10.6% in the previous technique, but our technique reduces 56.5% of the test data, which is five times higher than the previous technique.

Table 4.2. Test data reduction in different number of fan-out scan chains.

Circuit	technique	Number of scan chains					
		4	8	12	16	20	32
s13207	FSCANF	46.5%	57.7%	**58.1%**	53.9%	51.3%	37.0%
	ILS	**35.9%**	35.0%	29.2%	10.3%	20.2%	8.6%
s15850	FSCANF	29.8%	**36.8%**	34.6%	33.0%	23.2%	-6.7%
	ILS	**25.9%**	21.5%	21.6%	19.1%	14.5%	16.3%
s38417	FSCANF	56.2%	69.2%	72.0%	73.9%	**74.8%**	73.3%
	ILS	55.2%	61.8%	**64.5%**	64.4%	60.9%	63.5%
s38584	FSCANF	20.4%	35.3%	37.5%	**38.1%**	33.5%	26.8%
	ILS	9.6%	**10.3%**	8.1%	10.1%	5.2%	6.7%
s5378	FSCANF	38.8%	**55.2%**	37.6%	23.2%	0.2%	
	ILS	**10.6%**	0.8%	0.8%	0.8%	0.8%	
s9234	FSCANF	27.0%	**31.2%**	17.2%	5.3%	N/A	
	ILS	**12.9%**	4.8%	3.0%	3.6%	3.6%	

Table 4.3. Test application time reduction in different number of fan-out scan chains.

Circuit	technique	Number of scan chains					
		4	8	12	16	20	32
s13207	FSCANF	47.2%	60.5%	64.3%	65.5%	68.4%	**74.6%**
	ILS	**35.9%**	35.0%	29.2%	10.3%	20.2%	8.6%
s15850	FSCANF	31.1%	42.3%	46.8%	52.9%	54.4%	**62.4%**
	ILS	**25.9%**	21.5%	21.6%	19.1%	14.5%	16.3%
s38417	FSCANF	56.4%	70.0%	73.6%	76.7%	78.8%	**82.4%**
	ILS	55.2%	61.8%	**64.5%**	64.4%	60.9%	63.5%
s38584	FSCANF	21.0%	37.5%	42.5%	46.5%	47.0%	**56.3%**
	ILS	9.6%	**10.3%**	8.1%	10.1%	5.2%	6.7%
s5378	FSCANF	41.8%	55.2%	62.6%	65.0%	**67.9%**	
	ILS	**10.6%**	0.8%	0.8%	0.8%	0.8%	
s9234	FSCANF	30.5%	44.5%	48.0%	54.1%	**59.1%**	
	ILS	**12.9%**	4.8%	3.0%	3.6%	3.6%	

Note that our data shown for the previous technique may not be the best representation of the technique since we used a different ATPG tool, and the scan chain order is different from theirs. However, our technique can still enhance the compression ratio of the previous technique as long as it has test patterns in the serial scan mode because we can compress those serial patterns by using prelude vectors.

5. Conclusion

The fan-out scan chain structure is attractive in terms of reducing test data and application time because one test vector can fill in multiple scan chains at the same time. However, its usage is limited because it cannot be used if any conflicts exist across the fan-out scan chains. Previous techniques overcome this limitation by connecting all scan chains in series. Our proposed technique avoids this extreme case of serializing the scan chains. The compression ratio is optimized for individual patterns based on the distribution of specified bits in the pattern. The experimental results show that our technique significantly reduces test data volume as well as test application time.

6. References

[1] K-J. Lee, J-J. Chen, and C-H. Huang, "Using a single input to support multiple scan chains," *Dig. Tech. Papers, 1998 IEEE/ACM Int. Conf. Computer-Aided Design,* Nov. 1998, pp. 74-78.

[2] I. Hamzaoglu and J. Patel, "Reducing test application time for full scan embedded cores," *Dig. Papers, 29th Int. Symp. Fault-Tolerant Comp.,* June 1999, pp. 260-267.

[3] S. Makar, "A layout-based approach for ordering scan chain flip-flops," *Proc. of the Int. Test Conf.,* October 1998, pp. 341-347

[4] A Pandey and J. Patel, "Reconfiguration technique for reducing test time and test data volume in Illinois scan architecture based designs," *Proc. of IEEE VLSI Test Symp.*, April 2002.