September 2003

UILU-ENG-03-2218
CRHC-03-09

ENHANCEMENT OF THE ILLINOIS SCAN ARCHITECTURE FOR MULTIPLE SCAN INPUTS AND TRANSITION FAULTS

Mihir Atul Shah

Coordinated Science Laboratory
1308 West Main Street, Urbana, IL 61801
University of Illinois at Urbana-Champaign

REPORT DOCUMENTATION PAGE

Form Approved
OMB NO. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
September 2003

3. REPORT TYPE AND DATES COVERED

TITLE AND SUBTITLE
Enhancement of the Illinois Scan Architecture for Multiple Scan Inputs and Transition Faults

5. FUNDING NUMBERS
SRC 99-TJ-717

AUTHOR(S)
Mihir Atul Shah

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(ES)
University of Illinois
Coordinated Science Laboratory
1308 W. Main St.
Urbana, IL 61801

8. PERFORMING ORGANIZATION REPORT NUMBER
UILU-ENG-03-2218
(CRHC-03-09)

SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Semiconductor Research Corp.
P.O. Box 12053
Research Triangle Park, NC 27709-2053

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

. SUPPLEMENTARY NOTES

2a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12 b. DISTRIBUTION CODE

3. ABSTRACT *(Maximum 200 words)*

As VLSI circuits continue to become more complex, the cost of testing becomes increasingly important. As transistor feature sizes become smaller and transistor density increases, the complexity of testing increases. This leads to an increase in the test time for test generation and the test application process, which increases the test cost. In this thesis, we present methods for the reduction of test cost by shortening test application time and reducing the volume of data that needs to be stored. We give an analysis of the Illinois Scan Architecture (ILS) and determine its effectiveness in reducing test data volume and test application time. We propose a new method for further reducing test data volume and test application time involving the grouping of scan chains, called Multiple Group ILS. In addition, we present several algorithms for the reduction of groups, which can be used for Multiple Group ILS. Lastly, we extend the ILS methodology for testing transition faults and provide an analysis.

4. SUBJECT TERMS ILS, shared scan-in, multiple groups, transition fault

15. NUMBER IF PAGES
73

16. PRICE CODE

7. SECURITY CLASSIFICATION OR REPORT
UNCLASSIFIED

18. SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

19. SECURITY CLASSIFICATION OF ABSTRACT
UNCLASSIFIED

20. LIMITATION OF ABSTRACT
UL

ENHANCEMENT OF THE ILLINOIS SCAN ARCHITECTURE FOR MULTIPLE SCAN INPUTS AND TRANSITION FAULTS

BY

MIHIR ATUL SHAH

B.S., University of Illinois at Urbana-Champaign, 2001

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2003

Urbana, Illinois

ABSTRACT

As VLSI circuits continue to become more complex, the cost of testing becomes increasingly important. As transistor feature sizes become smaller and transistor density increases, the complexity of testing increases. This leads to an increase in the test time for test generation and the test application process, which increases the test cost. In this thesis, we present methods for the reduction of test cost by shortening test application time and reducing the volume of data that needs to be stored. We give an analysis of the Illinois Scan Architecture (ILS) and determine its effectiveness in reducing test data volume and test application time. We propose a new method for further reducing test data volume and test application time involving the grouping of scan chains, called Multiple Group ILS. In addition, we present several algorithms for the reduction of groups, which can be used for Multiple Group ILS. Lastly, we extend the ILS methodology for testing transition faults and provide an analysis.

To my parents

ACKNOWLEDGMENTS

I would like to thank my adviser, Prof. Janak Patel, for all his help and guidance with my research, and also for pointing me in the right direction when I needed it. Also I am thankful of my family for supporting me and encouraging me to finish my work. Lastly, I am thankful for my friends for delaying my research and making me appreciate my stay here at the University of Illinois.

TABLE OF CONTENTS

CHAPTER PAGE

LIST OF TABLES

TABLE PAGE

LIST OF FIGURES

FIGURE PAGE

CHAPTER 1

INTRODUCTION

As VLSI circuits continue to become more complex, the cost of testing becomes increasingly important. Current trends in VLSI still uphold Moore's Law, which predicts that the number of transistors per chip will double every 18 to 24 months. This is due to the approximate 10.5% reduction of transitor feature sizes every year, which allows for a 22.1% per year increase in transistor density [1]. This increase in density leads to increasing complexity of the test generation and application process, which increases the the time for testing. In fact, it has been shown that automated test equipment (ATE) is not evolving as fast as the circuits they test. As a result, testing time is becoming even more expensive. Therefore, it is imperative that testing time be reduced in order to reduce overall testing cost. In testers with insufficient high-speed memory to store all test vectors, multiple loading of test patterns from slower memory is required. Thus, the amount of test data has an indirect impact on test time. In this thesis, we address the issue of reducing tests costs by shortening test application time and reducing the volume of data that needs to be stored.

With the growing complexity of circuits, structured test techniques have replaced functional testing as the only way to achieve high fault coverage while maintaining short tester and development times [2]. Full scan is a structured test technique that has been widely adopted by industry because of the reduction in test complexity that it achieves. This is accomplished by making every flip-flop in the circuit controllable and observable by adding hardware that, during scan mode, converts all flip-flops in the circuit into a shift register. Thus, the entire state of the circuit can be controlled by shifting in values, and similarly the state can be observed by shifting out values. For test generation, this converts the circuit from a difficult sequential problem into an easier combinational problem, at the expense of increasing test application time.

Even though it is an improvement over functional testing, the test application time for full scan is still too long for many circuits. In addition, large test data volume becomes an issue when considering tester memory depth. If the test data exceeds the tester depth, then slower storage devices will have to be accessed during test application, further lengthening the test time. Also, if more faults are considered than just stuck-at testing, such as transition fault testing, then test data volume becomes even more significant.

Many improvements have been suggested to reduce test data volume and application time. One such method is using built-in-self-test (BIST), which provides on-chip test pattern generation and output comparison. In order to achieve sufficient fault coverage, using BIST alone is normally insufficient, and must be combined with standard scan approaches. Several proposed methods [3, 4, 5] have shown that using a hybrid of test pattern generation (ATPG) and BIST are effective in reducing tester data volume while still maintaining high fault coverage. Implementing BIST, however, introduces complications. There is an increase in area and routing when adding the BIST controller. Logic and signal changes/additions are needed to accommodate the additional hardware. All these factors greatly increase the complexity of the design, which leads to an increase in the development time of the chip [2].

An alternative to BIST is the Illinois Scan Architecture (ILS) [6]. Originally proposed as a technique for embedded cores, ILS can be used on standalone chips as well. Under the ILS methodology, there are two modes, broadcast and serial. In broadcast mode, the entire scan chain of the circuit is broken up into smaller scan chains, which all get scanned in with identical data supplied on a single pin. In serial mode, conventional scan patterns are created to test any faults not covered by broadcast mode. A case study of the Illinois Scan Architecture [2] on an industrial circuit showed that because of the parallelism involved in broadcast mode, there is a significant reduction in test data volume and application time, while introducing less of an area and routing penalty than other BIST implementations.

Other methods have been proposed to improve the original ILS methodology. An incremental algorithm [7] for ILS test generation was proposed which created an efficient way for finding the most optimal ILS configuration. A reconfigurable technique used with ILS was shown to reduce test data volume by decreasing the number of patterns applied in serial mode [8]. In this thesis, we propose a new technique involving ILS to further reduce test data volume by further eliminating the need for serial patterns, accomplished by the intelligent use of multiple groups of scan chains. Previous studies [2, 9] have demonstrated that using multiple groups instead of the one group used in traditional ILS broadcast mode can be beneficial, but the groups were predetermined without using any compatibility analysis.

Recently, a reconfigurable scan-chain architecture [10] was proposed which used compatibility analysis of scan chains for the assignment of groups. The algorithm worked iteratively, assigning each scan chain to one out of a fixed number of groups, then finding test patterns for that configuration for all undetectable faults, and running compatibility analysis on the remaining undetectable faults and then reassigning groups and testing, until all faults were detected. While it was shown that there was a significant reduction in data volume, there are some drawbacks to this method. First, because several configurations are needed for the circuit, there is extensive usage of the test generator. In addition, one method of compatibility analysis [11] involves making several ATPG runs to identify compatibility relations, thus making the use of the test generator even more extensive. Another drawback is that the other method of compatibility analysis used [12] assumes that a set of precomputed test patterns are available, which is not always the case. Our new method of using ILS with groups only uses one or two configurations, and does not need any precomputed patterns or separate ATPG runs for compatibility analysis.

Studies [13, 14] have shown that stuck-at faults may not be sufficient enough to cover all defects. As circuit time becomes more crucial, it may be necessary to perform additional tests to cover delay defects. Path delay tests are effective but hard to generate for full coverage because of the large size of circuits and their large path sets. An easier alternative is to use transition fault testing. Although these tests are not effective for all delay defects, they are useful in testing localized delay defects of large amounts of delay. Also, transition tests are easy to generate as they require only minimal modifications to a stuck-at fault test generator. While several methods [15, 16] have been proposed to create test generators for transition faults, little work has been done to reduce the test data volume created by these generators. In this thesis, we will apply the ILS techniques to transition faults to reduce data volume and test application time. ILS with transition faults is described in [2] using the skewed load technique [17]. In our testing we will be using the functional justification technique, described in [15].

1.1 Organization of the Thesis

The rest of this thesis is organized as follows. An introduction to Illinois Scan Architecture is presented in Chapter 2. The new method for test generation involving multiple scan chain groups, called Multiple Group ILS, is presented in Chapter 3. Algorithms for the reduction of groups using Multiple Group ILS are described in Chapter 4. Using ILS for transition fault testing is presented in Chapter 5. Finally, Chapter 6 discusses the conclusions and directions for future research.

CHAPTER 2

INTRODUCTION TO ILLINOIS SCAN ARCHITECTURE

The Illinois Scan Architecture (ILS) was first introduced in [6]. An overview of the architecture is provided here.

The architecture is shown in Figure 2.1, which represents the two modes of operation that it consists of. The top part of the figure shows a regular scan chain, which is known as Serial Mode. The bottom part of the figure shows the scan chain broken up into segments, called the Broadcast Mode. Here the scan-in pin that originally went into the entire scan chain now feeds into each of the scan chain segments. Thus, each segment will be inputted the same data in parallel. Note this shared scan-in idea has been reported in previous work [18]; however, this was limited to testing several independent circuits in parallel. As shown by the figure, the outputs of the scan chains are compressed

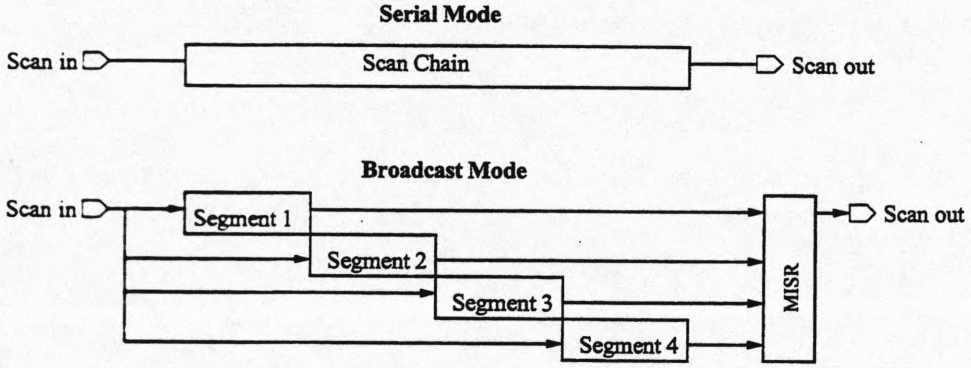


Figure 2.1 Two Modes of Illinois Scan Architecture

into a multiple input signature register (MISR). This is similar to the output response calculation found in BIST implementations. Like BIST implementations, in order to prevent corrupting the MISR signature, certain design rules must be followed, such as avoiding unknown states, internal bus conflicts, among others.

The additional hardware required for this architecture includes multiplexers for each scan chain segment, which are needed to switch between the two modes of operation. In addition, a MISR is needed with a size equal to the number of scan chain segments. It was shown that even with pessimistic assumptions, the addition of this hardware and the additional routing needed only caused a minor increase in the design area of a chip, less than what is needed for previously reported BIST implementations [2]. Also it is important to note that no additional test pins are required other than the ones already used for full scan.

2.1 Test Generation Procedure for ILS

As mentioned previously, the ILS technique requires two modes of operation, broadcast and serial. First test generation is performed in broadcast mode. However, because all the scan chain segments are receiving the same data, there are many constraints which are added on the test patterns, which make many faults untestable. Serial mode is then used to generate tests for all the undetectable faults in broadcast mode. This is because in this mode, there are no constraints on the test patterns and thus full fault coverage will be achieved.

(1) Generate a test set B under Broadcast Mode. Perform static compaction on B. Identify the set of faults, U, that are undetectable.

(2) Generate a test set S under Serial Mode targeting only the faults in U. Perform static compaction on S.

(3) Fault simulate test set S for all faults, and remove the detected faults from the complete fault list.

(4) Perform static compaction on test set B using the remaining faults from the previous step or regenerate B under Broadcast Mode using only these faults.

(5) Output the final test set T = B ∪ S.

Figure 2.2 Test Generation Procedure for ILS

The procedure we used for test generation with ILS is outlined by the steps in Figure 2.2. Note that although the serial patterns are only produced for the undetectable faults in broadcast mode, they also detect many more faults. Therefore, as shown in steps 3 and 4, all faults detected by serial mode are removed from the complete fault list and broadcast patterns are re-generated or compacted using only the remaining faults. Test pattern generation was accomplished by using ATOM [19], a robust combinational circuit ATPG. Static compaction was performed using a double detection and reverse order fault simulation technique [20].

2.2 Experimental Results

The test procedure mentioned above was performed using some of the ISCAS 89 benchmark [21] circuits. Table 2.1 shows the characteristics of the tested circuits. The columns in this table represent the name of the circuit, the number of primary inputs, the number of primary outputs, the number of flip-flops, the number of gates, the total amount of collapsed faults, and the total number of detectable faults, respectively.

Table 2.1 Characteristics of Tested Circuits

Circuit	Inputs	Outputs	Flip-Flops	Gates	Total Faults	Detectable Faults
s13207.1	62	152	638	7951	9815	9664
s15850.1	77	150	534	9772	11 725	11 336
s38417	28	106	1636	22 179	31 180	31 015
s38584.1	38	304	1426	19 253	36 303	34 797

First, test generation using the conventional full scan version of each circuit was conducted, and then static compaction was performed. All runs were performed on an AMD 900 MHz machine with 256 MB of memory. In order to compare the usefulness of the ILS architecture, test application time and test data volume needed to be measured. For a full scan circuit, it was assumed that the parallel access technique is used to access primary inputs and outputs of the circuit. During this process, the first vector is shifted in, and then after that shifting in a new vector and shifting out the response are performed simultaneously. Then the test application time, measured by the number of test cycles, is computed as

$$F + (1 + F) * V$$

where F is the number of flip-flops and V is the number of vectors in the test set. It is assumed that only one scan pin is used in the test application process. The test data volume for full scan circuits is calculated as

$$(PI + F) * V$$

where PI is the number of primary inputs, F is the number of flip-flops, and V is the number of test vectors. For comparison purposes with ILS configurations, we assume the outputs are compressed, and thus they are not included in the calculation for data volume. Table 2.2 shows the results of test generation on the full scan circuits. The columns in this table represent the circuit name, the number of test vectors after using the ATPG and performing static compaction, the number of cycles needed to test the circuit, the number of bits needed to be stored in the tester, and the total test time including both the test generator and static compaction.

Table 2.2 Results for Full Scan Circuits

Circuit	Test Vectors	Test Cycles	Test Data (bits)	Total Time (s)
s13207.1	468	299 690	327 600	9.6
s15850.1	432	231 654	263 952	11.0
s38417	921	1 509 313	1 532 544	40.7
s38584.1	634	906 144	928 176	34.5

Next, test generation is performed for the circuits using various ILS configurations. It is necessary first to describe some definitions related to the ILS architecture. For a given circuit, let the flip-flops be numbered 1,2,..., N, as given in the conventional scan chain configuration. For an ILS-k configuration, the scan chain is divided into smaller segments, each with a length of k flops. Because divisions are not necessarily perfect, the last scan chain may have a shorter length. The ILS-k configuration is illustrated in Figure 2.3. There are a total of m scan chain segments, where m - 1 of them are of length k, while the last chain segment is less than length k. Every segment receives identical data in parallel. For example, if a scan chain of length 100 were divided into 5 scan chain segments, flops 1, 21, 41, 61, and 81 would all receive the same data, since they are in the same position for every segment.

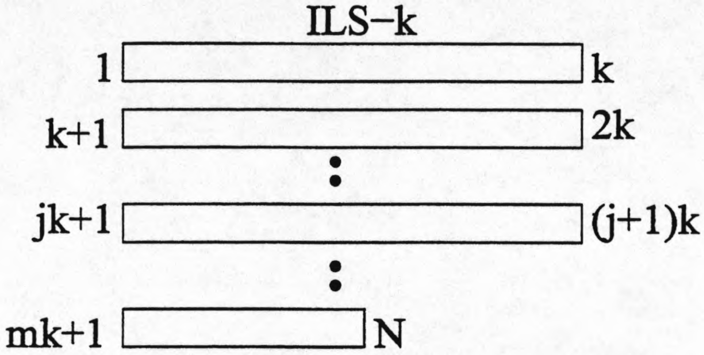


Figure 2.3 ILS-k Configuration

The test procedure for the ILS circuit was described in the preceding section. Various configurations for each circuit were tested in order to find the configuration that produced the most optimal results. ILS configurations were made by modifying the netlist of the full scan circuit, by connecting all scan chain segments to a single scan-in pin. The longest overall runtime for this procedure was approximately 10 min on 12 different configurations of circuit s38584.1. Table 2.3 shows experimental results for circuit s38584.1. The columns represent the ILS configuration, the number of scan chains, the number of broadcast patterns before reverse-order fault simulation (RFS), the fault coverage of the broadcast patterns, the additional faults that are undetectable in broadcast mode, the serial patterns needed to test those additional faults, the fault coverage of the serial patterns, the amount of broadcast vectors after RFS, and finally the fault coverage of these remaining broadcast patterns. 'Baseline' in column 1 refers to the full scan version of the circuit. From this table it is evident that, in general, as the length of the chain (column 1) decreases and the number of scan chain segments (column 2) increases, the number of broadcast patterns (column 3) decreases. This is due to the fact that as there are more parallel scan chain segments, there are tighter constraints on the ATPG because of increased redundancies, which cause many more undetectable faults, as shown by column 5. Since there are fewer detectable faults, fewer vectors are needed to test them. As the number of undetectable faults go up, more serial vectors will be needed to test them,

Table 2.3 Results for Various ILS Configurations for Circuit s38584.1

	Broadcast Mode (Pre-RFS)				**Serial Mode**		**Broadcast Mode (Post-RFS)**	
Config	**Num Chains**	**Patterns**	**FC%**	**Add Red**	**Patterns**	**FC%**	**Patterns**	**FC%**
Baseline	1	0	0	0	634	95.85	0	0
ILS-1024	2	946	95.83	8	2	46.76	618	49.09
ILS-800	2	918	95.75	36	17	63.57	616	32.28
ILS-512	3	956	95.71	51	25	67.45	626	28.40
ILS-400	4	957	95.62	84	32	69.57	605	26.28
ILS-256	6	936	95.34	187	52	77.43	623	18.42
ILS-200	8	863	95.04	293	99	86.80	534	9.05
ILS-128	12	902	94.47	503	100	85.12	564	10.73
ILS-100	15	846	94.27	576	162	89.44	481	6.41
ILS-64	23	726	91.69	1515	254	92.08	401	3.77
ILS-50	29	770	93.05	1017	247	91.66	406	4.19
ILS-32	45	655	89.69	2239	342	93.54	309	2.31
ILS-25	58	606	89.87	2172	367	93.76	281	2.09

shown in column 6. Since the serial vectors will test many more faults than what they are needed for, reverse-order fault simulation of the serial vectors and compaction eliminates many of the broadcast vectors that are now redundant. This form of compaction proves to be very effective, as shown by the comparison of Pre-RFS broadcast patterns (column 3) to the Post-RFS broadcast patterns (column 8).

When the ILS configuration is in broadcast mode, the tester only needs to shift in and shift out the length of the longest chain segment, since all the segments are being accessed in parallel. Thus the test application time, or test cycles, for broadcast mode is computed as

$$F_{LSC} + (1 + F_{LSC}) * V_B$$

where F_{LSC} is the length of the longest segment and V_B is the number of broadcast vectors required. Similarly, due to the parallel access of the ILS configuration, the tester does not need to store data for all the flops, but only for the length of the longest scan chain segment. The test data volume for broadcast mode is then computed as

$$(PI + F_{LSC}) * V_B$$

where PI is the number of primary inputs, F_{LSC} is the length of the longest chain in Broadcast Mode, and V_B is the number of broadcast vectors required. Note the outputs for an ILS configuration are compressed through a MISR, and thus they do not need to be specifically stored.

Table 2.4 shows the experimental results for the test application time required for various ILS configurations of all tested circuits. The columns in these table represent the circuit being tested, the ILS configuration, the number of serial patterns required, the tester cycles required for these serial patterns, the number of scan chain segments in broadcast mode, the final number of broadcast patterns needed after RFS, the tester cycles required to apply these patterns, the total tester cycles required including serial and broadcast mode, and finally the test application time reduction factor. The reduction factor is calculated as the ratio of the total cycles needed for the conventional full scan circuit (indicated by 'Baseline') to the total cycles needed for a particular ILS configuration.

The experimental results for test data volume for various ILS configurations of the tested circuits are shown in Table 2.5. The columns in this table represent the circuit being tested, the ILS configuration, the number of serial patterns required, the amount of data bits that needs to be stored in the tester for these serial patterns, the number of scan chain segments in broadcast mode, the final number of broadcast patterns needed after RFS, the amount of data bits that needs to be stored in the tester for these broadcast patterns, the total amount of data bits that needs to be stored for both broadcast and serial modes, and finally the reduction factor. The reduction factor is calculated as the ratio of the total bits needed to be stored for the full scan circuit to the total bits needed to be stored for a particular ILS configuration.

Table 2.4 Test Application Time Reduction Using ILS

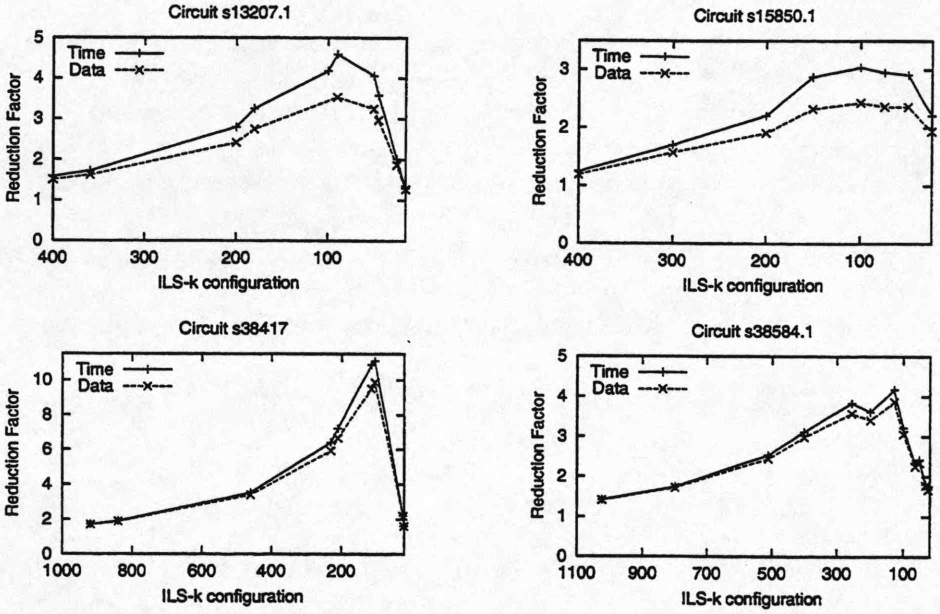
		Serial Mode		Broadcast Mode				
Circuit	**Config**	**Patterns**	**Test Cycles**	**Num Chains**	**Patterns**	**Test Cycles**	**Total Cycles**	**Reduction Factor**
s13207.1	Baseline	468	299 690	-	0	0	299 690	1.00
	ILS-400	3	2555	2	465	186 865	189 420	1.58
	ILS-360	9	6389	2	465	168 225	174 614	1.72
	ILS-200	25	16 613	4	449	90 449	107 062	2.80
	ILS-180	18	12 140	4	441	80 001	92 141	3.25
	ILS-100	44	28 754	7	425	43 025	71 779	4.18
	ILS-90	41	26 837	8	422	38 492	65 329	4.59
	ILS-50	85	54 953	13	370	18 920	73 873	4.06
	ILS-45	105	67 733	15	345	15 915	83 648	3.58
	ILS-25	221	141 857	26	237	6187	148 044	2.02
	ILS-15	359	230 039	43	81	1311	231 350	1.30
s15850.1	Baseline	432	231 654	-	0	0	231 654	1.00
	ILS-400	34	18 724	2	420	168 820	187 544	1.24
	ILS-300	19	10 699	2	416	125 516	136 215	1.70
	ILS-200	45	24 609	3	399	80 399	105 008	2.21
	ILS-150	35	192 59	4	406	61 456	80715	2.87
	ILS-100	72	39 054	6	366	37 066	76 120	3.04
	ILS-75	95	51 359	8	355	27 055	78 414	2.95
	ILS-50	118	63 664	11	314	16 064	79 728	2.91
	ILS-25	184	98 974	22	232	6057	10 5031	2.21
s38417	Baseline	921	1 509 313	-	0	0	1 509 313	1.00
	ILS-920	9	16 369	2	937	863 897	880 266	1.71
	ILS-840	11	19 643	2	920	774 560	794 203	1.90
	ILS-460	11	19 643	4	884	407 984	427 627	3.53
	ILS-230	15	26 191	8	905	209 285	235 476	6.41
	ILS-210	23	39 287	8	804	169 854	209 141	7.22
	ILS-115	21	36 013	15	883	102 543	138 556	10.89
	ILS-105	34	57 294	16	741	78 651	135 945	11.10
	ILS-23	405	664 621	72	456	10 967	675 588	2.23
	ILS-21	562	921 630	78	283	6247	927 877	1.63
s38584.1	Baseline	634	906 144	-	0	0	906 144	1.00
	ILS-1024	2	4280	2	618	634 474	638 754	1.42
	ILS-800	17	25 685	2	616	494 216	519 901	1.74
	ILS-512	25	37 101	3	626	321 650	358 751	2.53
	ILS-400	32	47 090	4	605	243 005	290 095	3.12
	ILS-256	52	75 630	6	623	160 367	235 997	3.84
	ILS-200	99	142 699	8	534	107534	250 233	3.62
	ILS-128	100	144 126	12	564	72 884	217 010	4.18
	ILS-100	162	232 600	15	481	48 681	281 281	3.22
	ILS-64	254	363 884	23	401	26 129	390 013	2.32
	ILS-50	247	353 895	29	406	20 756	374 651	2.42
	ILS-32	342	489 460	45	309	10 229	499 689	1.81
	ILS-25	367	525 135	58	281	7331	532 466	1.70

Table 2.5 Test Data Volume Reduction Using ILS

		Serial Mode		Broadcast Mode				
Circuit	Config	Patterns	Mem bits	Num Chains	Patterns	Mem bits	Total bits	Reduction Factor
s13207.1	Baseline	468	327 600	-	0	0	327 600	1.00
	ILS-400	3	2100	2	465	214 830	216 930	1.51
	ILS-360	9	6300	2	465	196 230	202 530	1.62
	ILS-200	25	17 500	4	449	117 638	135 138	2.42
	ILS-180	18	12 600	4	441	106 722	119 322	2.75
	ILS-100	44	30 800	7	425	68 850	99 650	3.29
	ILS-90	41	28 700	8	422	64 144	92 844	3.53
	ILS-50	85	59 500	13	370	41 440	100 940	3.25
	ILS-45	105	73 500	15	345	36 915	110 415	2.97
	ILS-25	221	154 700	26	237	20 619	175 319	1.87
	ILS-15	359	251 300	43	81	6237	257 537	1.27
s15850.1	Baseline	432	263 952	-	0	0	263 952	1.00
	ILS-400	34	20 774	2	420	200 340	221 114	1.19
	ILS-300	19	11 609	2	416	156 832	168 441	1.57
	ILS-200	45	27 495	3	399	110 523	138 018	1.91
	ILS-150	35	21 385	4	406	92 162	113 547	2.32
	ILS-100	72	43 992	6	366	64 782	108 774	2.43
	ILS-75	95	58 045	8	355	53 960	112 005	2.36
	ILS-50	118	72 098	11	314	39 878	111 976	2.36
	ILS-25	184	112 424	22	232	23 664	136 088	1.94
s38417	Baseline	921	1 532 544	-	0	0	1 532 544	1.00
	ILS-920	9	14 976	2	937	888 276	903 252	1.70
	ILS-840	11	18 304	2	920	798 560	816 864	1.88
	ILS-460	11	18 304	4	884	431 392	449 696	3.41
	ILS-230	15	24 960	8	905	233 490	258 450	5.93
	ILS-210	23	38 272	8	804	191 352	229 624	6.67
	ILS-115	21	34 944	15	883	126 269	161 213	9.51
	ILS-105	34	56 576	16	741	98 553	155 129	9.88
	ILS-23	405	673 920	72	456	23 256	697 176	2.20
	ILS-21	562	935 168	78	283	13 867	949 035	1.61
s38584.1	Baseline	634	928 176	-	0	0	928 176	1.00
	ILS-1024	2	2928	2	618	656 316	659 244	1.41
	ILS-800	17	24 888	2	616	516 208	541 096	1.72
	ILS-512	25	36 600	3	626	344 300	380 900	2.44
	ILS-400	32	46 848	4	605	264 990	311 838	2.98
	ILS-256	52	76 128	6	623	183 162	259 290	3.58
	ILS-200	99	144 936	8	534	127 092	272 028	3.41
	ILS-128	100	146 400	12	564	93 624	240 024	3.87
	ILS-100	162	237 168	15	481	66 378	303 546	3.06
	ILS-64	254	371 856	23	401	40 902	412 758	2.25
	ILS-50	247	361 608	29	406	35 728	397 336	2.34
	ILS-32	342	500 688	45	309	21 630	522 318	1.78
	ILS-25	367	537 288	58	281	17 703	554 991	1.67

It can be seen from both Tables 2.4 and 2.5 that as the length of the scan chains decrease and number of scan chain segments increase, the data volume and application time reduction factors start to increase, and then eventually end up decreasing. This is because as the length of the scan chains start to decrease, there are less flops that need to be shifted and stored, which causes the broadcast test cycles and memory bits to decrease. However, as the broadcast vectors start to decrease, more serial vectors are needed. Because serial vectors are costly in terms of both test application time and data volume, this increased cost starts to offset the reduction in cost from using smaller chain segments. Eventually, with a small enough scan chain length, the cost of needing serial vectors dominates over the cost of the broadcast vectors, which causes the reduction factor to decrease.

Figure 2.4 graphically shows the trend of reduction factor for test application time and test data volume versus the length of the longest scan chain segment. Both the application and data volume curves follow the same trends, since the only variable in both equations is the amount of flops that need to be shifted in/out, and stored, respectively. The maximum test application time reduction factor varies for every circuit, from 3.04 for circuit s15850.1 all the way to 11.1 for circuit s38417. The same is true for the maximum test data volume reduction factor, where values vary from 2.43 for circuit s15850.1 all the way to 9.88 for circuit s38417. For all cases, the maximum test application time reduction factor and the maximum test data volume reduction factor occurred using the same configuration. Although it is clear that there is a significant reduction in both test application time and test data volume for all tested circuits, it is not evident how to determine the optimal configuration without experimentation first.


Circuit s13207.1

Circuit s15850.1

Circuit s38417

Circuit s38584.1

Figure 2.4 Time and Data Reduction Factor versus ILS Configuration

2.3 Summary

In this chapter, the ILS architecture was implemented and investigated for several circuits, in order to determine its effectiveness. The results presented show a significant improvement in the reduction of both test application time and test data volume compared to the convention full scan technique.

CHAPTER 3

MULTIPLE GROUP ILS

In the preceding chapter, it was shown that using the ILS architecture is effective for both reducing the test application time and test data volume. This is accomplished by the use of broadcast mode, where several scan chains get shifted in identical data in parallel. However, because of this parallelism, there are increased redundancies when generating tests, which causes a loss in fault coverage, prompting the need for a serial mode which will then cover the remaining undetected faults. This mode is expensive both in terms of test application time and data volume. Therefore, a new method, called Multiple Group ILS, is proposed to eliminate the use of serial mode by using multiple groups of scan chain segments. Furthermore, we show that by using these groups, it is possible to eliminate broadcast mode as well.

3.1 Introductory Information

In order to understand the test procedure for Multiple Group ILS, it is first necessary to provide some background information related to the ILS architecture. In the following subsections, an example is given as an introduction and thereafter some definitions are provided.

3.1.1 Preliminary example

First, it is important to illustrate why certain faults become untestable for an ILS design in broadcast mode, as shown in Figure 3.1. The top portion shows an example of a partially specified serial mode pattern for a 12-flop circuit needed to test a certain fault that would be undetectable in broadcast mode. The bottom portion shows what this pattern would look like if it were 'folded over' and applied to the ILS structure in broadcast mode. Investigating the second column (the second bit position from the left for all the scan chains), scan chain one (SC1) needs a 0, while scan chain 2 (SC2) needs a 1. Since all the scan chains have the same scan-in pin, it is not possible to shift in both values at the same time, thus making this test vector inapplicable which in turn may make a fault untestable in broadcast mode. In general, for ILS designs in broadcast mode, a scan chain is said to be incompatible with another scan chain if, in the pattern needed to test a certain fault, there exists at least one bit position (column) where the

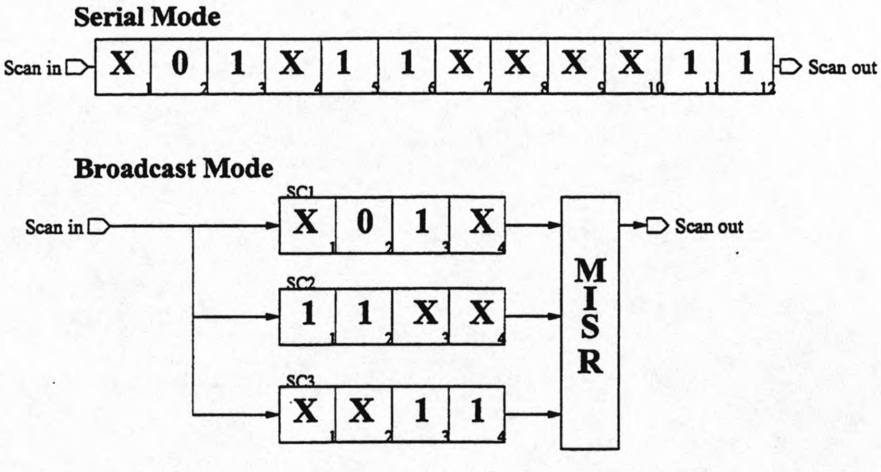


Figure 3.1 Example of an Untestable Fault in Broadcast Mode of ILS

two scan chains have conflicting binary values, i.e, one scan chain needs a one and the other needs a zero, or vice versa. In Figure 3.1, SC1 was incompatible with SC2 as they had conflicting values in the second bit position. Note that don't cares, marked by X's, do not conflict with any other values, and thus SC3 is compatible with both SC1 and SC2.

If two scan chains are incompatible with each other for a particular fault, then this fault will be untestable in broadcast mode. In the preceding chapter, serial mode was used to cover these untestable faults. In this chapter, a new solution is proposed, as depicted by the Figure 3.2. Here a second pin is introduced, which is connected to the bottom two scan chains, SC2 and SC3. Since SC1 and SC2 have different input pins, they are no longer incompatible, thus making the fault testable. SC2 and SC3 form one 'group' of scan chains, as they share the same scan-in pin, while SC1 is another group, and hence this arrangement is called Groups Mode. Note that while a pin is added on the input side, the output side remains the same as the conventional ILS configuration, with all scan chains connected to a MISR.

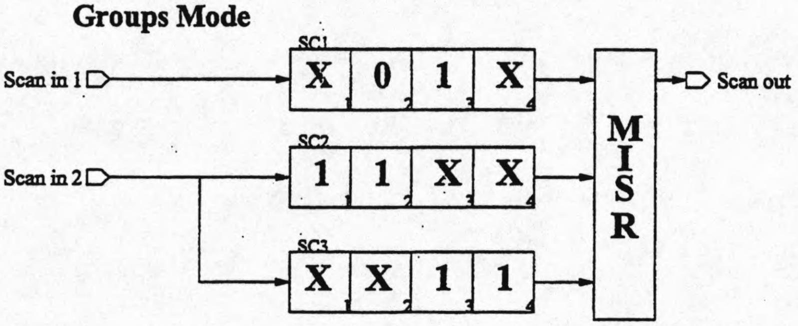


Figure 3.2 Previous Example Using Groups Mode

3.1.2 Compatibility definitions

In this subsection, we present some definitions that were introduced in the previous example.

(1) Two scan cells are said to be *compatible* if and only if no fault becomes untestable as a result of tying the two cells to a single input [11].

(2) Two scan chains are said to be *compatible* if and only if every pair of scan-cells that receive the same logic value are compatible [22].

Since the exact analysis for determining all pairwise compatibilities is computationally complex, we resort to a fast analysis which obtains a subset of all compatibilities. This analysis is based on a partially specified complete test set. This procedure was used as a first step in the more thorough procedures of [11] and [22].

By folding a given test set on an ILS organization, one can determine if two scan chains are compatible by simply noting an absence of value conflicts. Any two chains not found to be compatible by this procedure are termed *potentially incompatible.* Since the test set based on compatibility analysis does not find *all* compatibilities, any incompatible pairs found can only be said to be potentially incompatible. A further analysis (e.g., another test set) may determine that potentially incompatible pairs are indeed compatible. In other words, our procedure finds a super-set of actual incompatibilities. It is sufficient (but not necessary) to remove these incompatibilities to achieve complete fault coverage. This is the basis of our procedure for altering the ILS structure such that all potential incompatibilities are removed. In subsequent sections, we will drop the adjective "potential" for incompatibilities.

3.2 Test Generation Procedure

In the example described in the previous section, a group configuration for scan chains was created for one particular fault. In this section, we describe the test procedure for creating the proper group configuration for testing *all* undetectable faults in broadcast mode for a given circuit. Figure 3.3 gives an outline of the test procedure used to generate patterns for Multiple Group ILS using both broadcast and groups modes. First, test generation for broadcast mode is performed. Then, for the remaining undetected faults, an ATPG without random-fill is used to generate serial patterns. Compatibility analysis of these serial patterns is then performed, which is detailed in the following subsection. After the correct group configuration has been found, we perform test generation for groups mode for only the undetectable faults in broadcast mode. Since groups mode will test more faults than what it is needed for, reverse-order fault simulation (RFS) for groups mode is then performed and all detected faults are removed from the complete fault list. Test patterns for broadcast mode are then re-generated, or compacted, using this new fault list.

Test pattern generation with random-fill was performed by using ATOM [19], while generation without random-fill was accomplished by using a modified version of ATOM. Static compaction for fully specified patterns was performed using a double detection and reverse order fault simulation technique [20], while compaction for partially specified

(1) Regenerate the netlist for broadcast mode. Generate a test set B under Broadcast Mode. Perform static compaction on B. Identify the set of faults, U, that are undetectable.

(2) Regenerate the netlist for serial mode. Using a *non-filling* ATPG, generate a partially specified test set S tagerting only the faults in U. Perform static compaction on S to remove redundant vectors.

(3) Perform compatability analysis for all patterns in S to find a group configuration.

(4) Regenerate the netlist for groups mode. Generate a test set G under Groups Mode targeting only the faults in U. Perform static compaction on G.

(5) Fault simulate test set G for all faults, and remove the detected faults from the complete fault list.

(6) Regenerate the netlist for broadcast mode. Perform static compaction on test set B using the remaining faults from the previous step or regenerate B under Broadcast Mode using only these faults.

(7) Output the final test set T = B ∪ G.

Figure 3.3 Test Procedure for Using Broadcast and Groups Mode

patterns was performed using a simpler forward and reverse order single detection scheme [1].

3.2.1 Compatibility analysis

In order to determine the optimal configuration of groups, it is necessary to perform compatibility analysis on the scan chains. In general, the objective is to form the minimal amount of groups so that all scan chains within any group are compatible, as this will result in the least amount of test data volume that needs to be stored. To accomplish this, first an incompatibility graph, with all the scan chains as nodes, needs to be created from the serial patterns generated by the non-random-fill ATPG. It is essential not to have any randomly-filled inputs as this will lead to extraneous incompatibilities between scan chains. The process is simple: like the example in the previous section, for each serial pattern generated, we 'fold over' the pattern into the scan chains used in broadcast mode and for every bit position, determine if there are conflicting values. When a conflict arises, we note which chains are conflicting, and add an edge between the two scan chains, represented as nodes in the incompatibility graph. Thus for the example in the previous section, the graph would look as shown in Figure 3.4, which depicts that SC1 is incompatible with SC2, while SC3 has no incompatibilities.

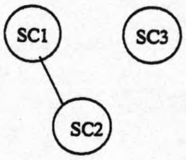


Figure 3.4 Incompatibility Graph for Example Shown in Figure 3.1

Note that the same graph is used for all the serial patterns generated; edges are added between scan chains if new incompatibilities are found.

After the incompatibility graph has been created, a graph coloring algorithm is applied, which will assign a specific color to every node such that no two nodes connected by an edge will have the same color. Here, assigning colors to nodes is equivalent to assigning a group number, or scan-in pin, to a scan chain. The graph coloring algorithm will attempt to assign the minimal number of colors (groups) while following the restriction that two adjacent nodes (two incompatible scan chains) cannot be of the same color (group number). For the example shown in Figure 3.4, SC1 and SC2 have to be different colors, while SC3 can be any color. Thus a minimum of two colors are needed, which corresponds to a minimum of two scan-in pins, or groups needed for testing.

Although in the preceding example it was trivial to determine the coloring, in general determining the optimal coloring for a graph with many edges can be a difficult problem. For the incompatibility graphs generated in our procedure, a graph coloring algorithm based upon the DSATUR algorithm [23] was utilized. The algorithm first attempts to find the maximum clique size in the graph, which corresponds to finding the maximum number of nodes that are all connected to each other. For example, in the graph in Figure 3.5, the maximum clique is of size 3, since SC1, SC2, and SC3 are all interconnected.

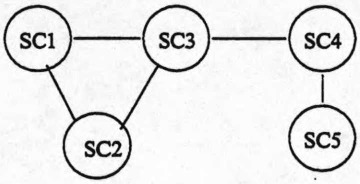


Figure 3.5 Incompatibility Graph with Maximum Clique of Size 3 (SC1,SC2,SC3)

This maximum clique number then represents the lower bound of the number of colors needed, as every node in that clique needs to be a different color, since they are all interconnected. Those nodes are then colored first, as suggested in [24].

Then a heuristic method is applied for the rest of the graph. A simple explanation of the algorithm is provided here. For more details on the exact algorithm, see [23, 25, 26]. The algorithm works by looking at node in the graph with the highest *saturation degree* first, which is defined as the number of different colors found for all adjacent nodes. If more than one node shares this highest saturation degree, then the algorithm attempts to find the node with the highest unlabeled degree, defined as the largest number of uncolored adjacent nodes. If there is still more than one node left, then one is chosen randomly. After a node is chosen, it is assigned the lowest feasible color, depending on the color of its adjacent nodes. This process is repeated until all nodes are colored. This will result in a configuration of groups for which there are no incompatibilities, which will be then be used for groups mode.

3.3 Experimental Results

The test procedure was performed on the same circuits from the ISCAS 89 benchmark described in the preceding chapter. All tests were run on the same machine as well, an AMD 900 MHz with 256 MB of memory. The longest overall run time was approximately 35 min for the 10 configurations of circuit s38417. This time is longer than the traditional ILS procedure because of a few reasons. First, smaller chain length configurations are used, which causes the ATPG to run longer as there are more redundancies in the circuit. Second, extra time is needed for compatibility analysis and the extra serial pattern ATPG run required for this analysis. In addition, compaction for partially specified vectors takes more time than compaction for fully specified vectors.

Table 3.1 shows experimental results for the tested circuits. The columns represent the circuit being tested, the ILS configuration, the number of scan chains, the number of broadcast mode patterns generated before reverse-order fault simulation (RFS), the additional undetectable faults caused by this ILS configuration, the number of vectors created by the nonfilling ATPG to cover these undetectable faults, the number of groups found after compatibility analysis, the number of groups mode vectors, and finally the number of broadcast mode patterns remaining after fault simulation of the groups mode patterns.

Table 3.1 Results for Multiple Group ILS on Various Configurations

Circuit	Config	Num Chains	Pre-RFS Broadcast Vectors	Addnl Red Faults	Num Topoff Vectors	Num Groups	Num Group Vectors	Post-RFS Broadcast Vectors
s13207.1	ILS-100	7	588	211	57	5	38	427
	ILS-90	8	588	100	63	4	40	422
	ILS-75	9	588	126	88	5	145	311
	ILS-45	15	568	209	130	5	94	351
	ILS-25	26	323	1969	890	6	223	236
	ILS-15	43	239	1909	852	7	350	86
	ILS-10	64	155	2505	1110	8	385	55
	ILS-6	107	69	3054	1311	9	425	18
s15850.1	ILS-125	5	504	148	74	4	52	369
	ILS-100	6	519	122	87	5	71	366
	ILS-75	8	469	178	116	6	93	350
	ILS-50	11	475	279	161	7	119	318
	ILS-25	22	412	558	278	6	190	232
	ILS-15	36	339	1011	444	6	250	162
	ILS-10	54	275	1596	604	7	303	115
	ILS-6	89	243	2016	721	11	315	98
s38417	ILS-210	8	1191	40	32	2	20	803
	ILS-115	15	1266	45	35	3	25	879
	ILS-105	16	1127	52	44	2	28	744
	ILS-75	22	1138	324	186	3	80	694
	ILS-64	26	1248	500	231	3	115	745
	ILS-32	52	1048	1066	537	5	252	594
	ILS-26	63	981	1273	685	6	316	540
	ILS-20	82	705	2021	992	7	379	344
	ILS-16	103	672	3113	1433	9	602	305
	ILS-14	117	398	3827	1713	10	601	186
s38584.1	ILS-200	8	863	293	195	5	103	547
	ILS-128	12	902	503	310	5	103	572
	ILS-100	15	846	576	374	7	162	493
	ILS-64	23	726	1513	914	6	262	398
	ILS-50	29	770	1017	650	5	238	413
	ILS-32	45	655	2239	1311	6	345	313
	ILS-25	58	606	2170	1316	6	366	282
	ILS-20	72	551	2681	1631	7	396	241
	ILS-16	90	458	3883	2267	7	455	155
	ILS-12	119	341	4990	2886	8	509	105

From this table, it is evident that for all circuit, as the length of the scan chain (column 2) becomes shorter, and the number of chains (column 3) increases, more faults become undetectable (column 5). This causes the number of broadcast vectors (column 3) to decrease, and also increases the number of 'top-off' vectors (column 6) needed to cover these undetectable faults. In general, as there are more top-off vectors and larger numbers of scan chains, this will cause more incompatibilities between the scan chains. This results in more groups being formed (column 7). As the scan chain segments become shorter and there is an increase in undetectable faults, more group vectors (column 8) are needed to detect them. This will result in even less broadcast vectors (column 9), as the group vectors will be able to detect many of the faults in the complete fault list.

The equations to compute test application time and test data volume for broadcast mode are described in the previous chapter. For ILS configurations in groups mode, the test application time calculation is the same as broadcast mode, as all scan chains are still getting inputted vectors in parallel. The test application time for groups mode is then computed as:

$$F_{LSC} + (1 + F_{LSC}) * V_G$$

where F_{LSC} is the length of the longest segment and V_G is the number of groups mode vectors required. For test data volume calculation, there will be more input pins than broadcast mode, and thus more data will need tc be stored in the tester. The test data volume for groups mode is calculated as:

$$(PI + F_{LSC} * N_G) * V_G$$

where PI is the number of primary inputs, F_{LSC} is the length of the longest chain in groups mode, N_G is the number of groups (or scan-in pins required), and V_G is the number of test vectors needed for groups mode.

Table 3.2 shows the experimental results for test data volume needed for various configurations of the four tested circuits. The columns in this table represent the circuit being tested, the ILS configuration, the number of groups used for groups mode, the number of groups mode patterns required, the amount of test data that needs to be stored for these patterns, the number of scan chain segments in broadcast mode, the final number of broadcast patterns needed after RFS, the amount of test data needed to be stored to apply these patterns, the total test data needed to be stored including groups and broadcast modes, and finally the data volume reduction factor. The reduction factor is calculated as the ratio of the test data volume needed for the conventional full scan circuit (indicated by 'Baseline*') to the test data volume needed for a particular ILS configuration.

The results from this table indicate that for all circuits, there is a far more significant reduction in test data volume than for the traditional ILS procedure. This is due to the fact that groups mode has replaced serial mode. In terms of data volume, using serial mode is the equivalent as having one scan-in pin, or group, for each scan chain.

Table 3.2 Test Data Volume Reduction Using Multiple Group ILS

		Group Mode			Broadcast Mode				
Circuit	**Config**	**Num Groups**	**Num Pats.**	**Mem bits**	**Num Chains**	**Num Pats.**	**Mem bits**	**Total bits**	**Red Factor**
s13207.1	Baseline*	-	468*	327 600	-	0	0	327 600	1.00
	ILS-100	5	38	21 356	7	427	69 174	90 530	3.62
	ILS-90	4	40	16 880	8	422	64 144	81 024	4.04
	ILS-75	5	145	63 365	9	311	42 607	105 972	3.09
	ILS-45	5	94	26 978	15	351	37 557	64 535	5.08
	ILS-25	6	223	47 276	26	236	20 532	67 808	4.83
	ILS-15	7	350	58 450	43	86	6622	65 072	5.03
	ILS-10	8	385	54 670	64	55	3960	58 630	5.59
	ILS-6	9	425	49 300	107	18	1224	50 524	6.48
s15850.1	Baseline*	-	432*	253 952	-	0	0	253 952	1.00
	ILS-125	4	52	30 004	5	369	74 538	104 542	2.43
	ILS-100	5	71	40 967	6	366	64 782	105 749	2.40
	ILS-75	6	93	49 011	8	350	53 200	102 211	2.48
	ILS-50	7	119	50 813	11	318	40 386	91 199	2.78
	ILS-25	6	190	43 130	22	232	23 664	66 794	3.80
	ILS-15	6	250	41 750	36	162	14 904	56 654	4.48
	ILS-10	7	303	44 541	54	115	10 005	54 546	4.66
	ILS-6	11	315	45 045	89	98	8134	53 179	4.78
s38417	Baseline*	-	921*	1 532 544	-	0	0	1 532 544	1.00
	ILS-210	2	20	8960	8	803	191 114	200 074	7.66
	ILS-115	3	25	9325	15	879	125 697	135 022	11.35
	ILS-105	2	28	6664	16	744	98 952	105 616	14.51
	ILS-75	3	80	20 240	22	694	71 482	91 722	16.71
	ILS-64	3	115	25 300	26	745	68 540	93 840	16.33
	ILS-32	5	252	47 376	52	594	35 640	83 016	18.46
	ILS-26	6	316	58 144	63	540	29 160	87 304	17.55
	ILS-20	7	379	63 672	82	344	16 512	80 184	19.11
	ILS-16	9	602	103 544	103	305	13 420	116 964	13.10
	ILS-14	10	601	100 968	117	186	7812	108 780	14.09
s38584.1	Baseline*	-	634*	928 176	-	0	0	928 176	1.00
	ILS-200	5	103	106 914	8	547	130 186	237 100	3.91
	ILS-128	5	103	69 834	12	572	94 952	164 786	5.63
	ILS-100	7	162	119 556	15	493	68 034	187 590	4.95
	ILS-64	6	262	110 564	23	398	40 596	151 160	6.14
	ILS-50	5	238	68 544	29	413	36 344	104 888	8.85
	ILS-32	6	345	79 350	45	313	21 910	101 260	9.17
	ILS-25	6	366	68 808	58	282	17 766	86 574	10.72
	ILS-20	7	396	70 488	72	241	13 978	84 466	10.99
	ILS-16	7	455	68 250	90	155	8370	76 620	12.11
	ILS-12	8	509	68 206	119	105	5250	73 456	12.64

*** indicates Serial Mode**

Thus, as long as there are less groups than scan chains, groups mode will require less data to be stored than serial mode. This is most prevalent when using the smallest chain lengths, as there is a much larger difference between the number of groups and the number of scan chains. As a result, the maximum data volume reduction factor is found with smaller chain lengths. For larger industrial circuits, this will be beneficial in terms of tester depth. By using the optimal configuration using the Multiple Group ILS procedure, it is much less likely that test data will exceed tester scan channel depth, in which case access to slower mass storage would have been required.

Table 3.3 shows the experimental results for the test application time (tester cycles) needed for various configurations of the four tested circuits. The columns in this table represent the circuit being tested, the ILS configuration, the number of groups used for groups mode, the number of groups mode patterns required, the tester cycles required for these patterns, the number of scan chain segments in broadcast mode, the final number of broadcast patterns needed after RFS, the tester cycles required to apply these patterns, the total tester cycles required including groups and broadcast modes, and finally the test application time reduction factor. The reduction factor is calculated as the ratio of the total cycles needed for the conventional full scan circuit (indicated by 'Baseline*') to the total cycles needed for a particular ILS configuration.

It is clear from this table that the test application time is greatly reduced as compared to the traditional ILS configuration. This is due to the fact that in serial mode, values have to shifted in one at a time to every flop. In groups mode, values are shifted in parallel to every group, and thus there is a large reduction in the number of tester cycles.

Table 3.3 Test Application Time Reduction Using Multiple Group ILS

		Group Mode			Broadcast Mode				
Circuit	Config	Num Groups	Num Pats.	Tester Cycles	Num Chains	Num Pats.	Tester Cycles	Total Cycles	Red Factor
s13207.1	Baseline*	-	468*	299 690	-	0	0	299 690	1.00
	ILS-100	5	38	3938	7	427	43 227	47 165	6.35
	ILS-90	4	40	3730	8	422	38 492	42 222	7.10
	ILS-75	5	145	11 095	9	311	23 711	34 806	8.61
	ILS-45	5	94	4369	15	351	16 191	20 560	14.58
	ILS-25	6	223	5823	26	236	6161	11 984	25.01
	ILS-15	7	350	5615	43	86	1391	7006	42.78
	ILS-10	8	385	4245	64	55	615	4860	61.66
	ILS-6	9	425	2981	107	18	132	3113	96.27
s15850.1	Baseline*	-	432*	231 654	-	0	0	231 654	1.00
	ILS-125	4	52	6677	5	369	46 619	53 296	4.35
	ILS-100	5	71	7271	6	366	37 066	44 337	5.22
	ILS-75	6	93	7143	8	350	26 675	33 818	6.85
	ILS-50	7	119	6119	11	318	16 268	22 387	10.35
	ILS-25	6	190	4965	22	232	6057	11 022	21.02
	ILS-15	6	250	4015	36	162	2607	6622	34.98
	ILS-10	7	303	3343	54	115	1275	4618	50.16
	ILS-6	11	315	2211	89	98	692	2903	79.80
s38417	Baseline*	-	921*	1 509 313	-	0	0	1 509 313	1.00
	ILS-210	2	20	4430	8	803	169 643	174 073	8.67
	ILS-115	3	25	3015	15	879	102 079	105 094	14.36
	ILS-105	2	28	3073	16	744	78 969	82 042	18.40
	ILS-75	3	80	6155	22	694	52 819	58 974	25.59
	ILS-64	3	115	7539	26	745	48 489	56 028	26.94
	ILS-32	5	252	8348	52	594	19 634	27 982	53.94
	ILS-26	6	316	8558	63	540	14 606	23 164	65.16
	ILS-20	7	379	7979	82	344	7244	15 223	99.15
	ILS-16	9	602	10 250	103	305	5201	15 451	97.68
	ILS-14	10	601	9029	117	186	2804	11 833	127.55
s38584.1	Baseline*	-	634*	90 6144	-	0	0	906 144	1.00
	ILS-200	5	103	20 903	8	547	110 147	131 050	6.91
	ILS-128	5	103	13 415	12	572	73 916	87 331	10.38
	ILS-100	7	162	16 462	15	493	49 893	66 355	13.66
	ILS-64	6	262	17 094	23	398	25 934	43 028	21.06
	ILS-50	5	238	12 188	29	413	21 113	33 301	27.21
	ILS-32	6	345	11 417	45	313	10 361	21 778	41.61
	ILS-25	6	366	9541	58	282	7357	16 898	53.62
	ILS-20	7	396	8336	72	241	5081	13 417	67.54
	ILS-16	7	455	7751	90	155	2651	10 402	87.11
	ILS-12	8	509	6629	119	105	1377	8006	113.18

* indicates Serial Mode

Here, for every circuit the greatest reduction factor was with the smallest chain length possible. This is because the number of total vectors, including both groups mode and broadcast mode, remained roughly about the same, making the length of the longest scan chain segment the only variable in the computation for test application time.

For comparison purposes with the results in Chapter 2, it was assumed that only one pin was used for serial mode, even though several pins were used for groups mode. A more accurate calculation of the test application time reduction factor would depend on the number of scan-in pins a tester would be able to use. If a scan-in pin was available for every scan chain, then there would be no improvement in test application time. However, in larger industrial circuits with many scan chains, it is likely that there would not be scan-in pins available for every scan chain. Thus, using the Multiple Groups ILS procedure would still prove to be beneficial in these cases.

The fault coverage for all the tested circuits did not change in our experiments. Thus we were able to completely replace serial mode with groups mode. In this case, the additional hardware required would be the same as the traditional ILS procedure previously described. The structure may be different however, as shown by the example in Figure 3.6.

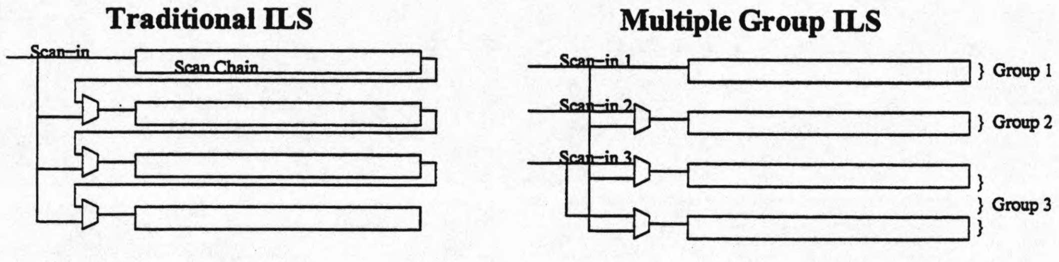


Figure 3.6 Structure of Traditional ILS versus Multiple Group ILS

It is possible that for larger circuits, there would be a loss in fault coverage because of aborted faults from trying to generate tests using smaller chain segments. If full fault coverage is then desired, serial mode could be added by using a three-input multiplexer instead of a two-input multiplexer.

3.4 Multiple Group ILS Using Only a Single Mode

In the previous section, it was shown that by using Multiple Group ILS with both broadcast and groups modes, a significant reduction in test data volume and application time was achieved. This was accomplished using the same hardware as the traditional ILS approach. In this section, we show that by sacrificing some reduction in test data volume, it is possible just to use groups mode alone for test application. This will eliminate the multiplexers needed for the traditional ILS approach, and also reduce routing. In addition, the use of only a single mode reduces the time for test generation compared to the dual mode procedure in the previous section.

The test procedure is the same as steps 1 through 4 in Figure 3.3, with the exception that in step 4, we perform test generation for *all* faults, not just the undetectable faults in broadcast mode. Since faults detected by broadcast mode were able to be detected with just one scan-in pin, it follows that they should be able to detected with any group configuration.

The results for the four tested circuits are shown in Table 3.4.

Table 3.4 Experimental Results for Multiple Group ILS Using a Single Mode

				Data Volume		Appl. Time	
		Num	Num	Mem	Red	Test	Red
Circuit	Config	Groups	Pats.	bits	Factor	Cycles	Factor
s13207.1	Baseline*	-	468*	327 600	1.00	299 690	1.00
	ILS-100	5	461	259 082	1.26	46 661	6.42
	ILS-90	4	480	202 560	1.62	43 770	6.85
	ILS-75	5	473	206 701	1.58	36 023	8.32
	ILS-45	5	463	132 881	2.47	21 343	14.04
	ILS-25	6	471	99 852	3.28	12 271	24.42
	ILS-15	7	469	78 323	4.18	7519	39.86
	ILS-10	8	466	66 172	4.95	5136	58.35
	ILS-6	9	449	52 084	6.29	3149	95.17
s15850.1	Baseline*	-	432*	263 952	1.00	231 654	1.00
	ILS-125	4	439	253 303	1.04	55 439	4.18
	ILS-100	5	439	253 303	1.04	44 439	5.21
	ILS-75	6	438	230 826	1.14	33 363	6.94
	ILS-50	7	433	184 891	1.43	22 133	10.47
	ILS-25	6	443	100 561	2.62	11 543	20.07
	ILS-15	6	428	71 476	3.69	6863	33.75
	ILS-10	7	432	63 504	4.16	4762	48.65
	ILS-6	11	423	60 489	4.36	2967	78.08
s38417	Baseline*	-	921*	1 532 544	1.00	1 509 313	1.00
	ILS-210	2	891	399 168	3.84	188 211	8.02
	ILS-115	3	903	336 819	4.55	104 863	14.39
	ILS-105	2	850	202 300	7.58	90 205	16.73
	ILS-75	3	929	235 037	6.52	70 679	21.35
	ILS-64	3	890	195 800	7.83	57 914	26.06
	ILS-32	5	899	169 012	9.07	29 699	50.82
	ILS-26	6	884	162 656	9.42	23 894	63.17
	ILS-20	7	885	148 680	10.31	18 605	81.12
	ILS-16	9	918	157 896	9.71	15 622	96.61
	ILS-14	10	856	143 808	10.66	12 854	117.42
s38584.1	Baseline*	-	634*	928 176	1.00	906 144	1.00
	ILS-200	5	622	645 636	1.44	125 222	7.24
	ILS-128	5	637	431 886	2.15	82 301	11.01
	ILS-100	7	642	473 796	1.96	64 942	13.95
	ILS-64	6	631	266 282	3.49	41 079	22.06
	ILS-50	5	637	183 456	5.06	32 537	27.85
	ILS-32	6	638	146 740	6.33	21 086	42.97
	ILS-25	6	640	120 320	7.71	16 665	54.37
	ILS-20	7	634	112 852	8.22	13 334	67.96
	ILS-16	7	629	94 350	9.84	10 709	84.62
	ILS-12	8	634	84 956	10.93	8254	109.78

* indicates Serial Mode

The columns in this table represent the circuit being tested, the ILS configuration used, the number of groups found after compatibility analysis, the number of patterns generated, the amount of memory bits needed to be stored in the tester, the test data volume reduction factor, the amount of cycles needed for testing, and finally the test application time factor. Reduction factors were calculated in the same manner as mentioned in the previous section. Investigation of this table reveals that the test data volume reduction is less than using broadcast and groups modes, however it is still a significant improvement over the traditional ILS approach. The test application time remained approximately the same as using dual modes, since all scan chains are still being shifted in/out data in parallel, and the total number of patterns remained approximately the same as with dual mode.

3.5 Summary

In this chapter, we presented a new procedure for ILS designs using multiple groups. Two methods were described: using both broadcast and groups modes and using just groups mode. Both methods were shown to have a significant improvement in both test data volume and test application time reduction over the traditional ILS approach.

CHAPTER 4

GROUP REDUCTION FOR MULTIPLE GROUP ILS

In the previous chapter, we introduced a method for reducing test volume and test application time by using multiple scan-in pins and grouping scan chains together according to their compatibilities. In this chapter, we present several algorithms to reduce the number of groups that would be used in groups mode of Multiple Group ILS. All algorithms were implemented but did not show any significant improvement for the circuits we tested. However, for larger industrial circuits, these variations have the potential to produce better results, and thus are mentioned here in the following sections.

4.1 Addition of Inverse Chains for Compatibility Analysis

In the previous chapter, compatibility analysis was performed only using the standard noninverted scan chains. It is possible to further reduce the number of incompatibilities by also including the inverse of the scan chain as well. Then for each scan chain, we would then have to choose between using either the noninverted chain or the inverted chain. An additional inverter would be required for every inverted chain selected.

Figure 4.1 shows the procedure we implemented for compatibility analysis using inverted and noninverted chains.

(1) Create an incompatability graph so that every scan chain in the circuit has 2 nodes representing it, one for the inverted scan chain and one for the non-inverted scan chain.

(2) For a given set of partially specified test vectors, add incompatability edges accordingly for all nodes in the graph.

(3) Use the graph coloring algorithm to produce a group configuration for all nodes in the incompatability graph

(4) Investigate all groups created in order of largest size group (the one with the most amount of chains) first and inspect each node within the group. For each node, mark if the chain it represents has or hasn't been found yet in a previous group.

(5) Investigate all groups in order of smallest size group first. If all nodes within the group have been marked as previously found, then remove this group, by removing all nodes and edges associated with these nodes from the incompatability graph. Go to step 3. Else, if at least one node within the group hasn't been marked as previously found, proceed to the next smallest group. If every group has at least one node that hasn't been previously found, then proceed to step 6.

(6) Prune remaining nodes in the graph so that no chain has both its inverted and non-inverted nodes representing it. If this is the case, then the inverted node is removed.

Figure 4.1 Procedure for Compatibility Analysis Using Both Inverted and Noninverted Chains

For each scan chain, include both its inverted and noninverted version as nodes in the incompatibility graph. Then we color the graph including all the nodes, and remove the smallest group which is redundant. A group is redundant if, for all the nodes within the group, the chain the node represents has already been included in a larger size group. After removing the nodes of the redundant group from the incompatibility graph, we color the graph again and repeat the same process. This continues until there are no more groups to remove, and thus all groups have at least one node whose chain it represents is not found in any other group. Finally, the groups are pruned so every chain will have only either its noninverted or inverted node representing it.

After applying this procedure to the circuits and ILS configurations tested in the previous chapter, we found that in a majority of cases, the number of groups was equivalent to the number of groups found by only using the noninverted version of the chains. This is due to the fact that we are testing fairly small circuits with a small amount of scan chains. It is very possible that better results could be achieved with bigger circuits that would have more scan chains. Also using a better algorithm for selecting between inverted and noninverted chains could also improve results.

4.2 Edge Reduction of an Incompatibility Graph

For an incompatibility graph produced for an ILS configuration of a circuit, it is beneficial to have as few edges, or incompatibilities, between the chains as possible. In the previous section, it was mentioned that the incompatibility graph we generate represents only potential incompatibilities, and it is possible to refine the graph by reclassifying a potential incompatibility into an assumed compatibility. With fewer incompatibilities, there is a greater chance that less groups will be needed to 'color' the graph, which will result in less test data volume that needs to be stored. In the following subsections, we present three methods of edge reduction.

4.2.1 Edge reduction using compatibility relations

Edge reduction of an incompatibility graph is possible by looking at relations between ILS configurations. This idea is best explained by looking at Figure 4.2. The bottom portion shows an ILS-n configuration with 4 scan chains each of length n. The top portion shows an ILS-$2n$ configuration with 2 scan chains each of length $2n$. Then for the ILS-n configuration, if the compatibilities for every scan chain is combined with the next/previous scan chain, these combined chains then match the chains in the ILS-$2n$ configuration. It follows then that a compatibility between these 'extended' scan chains in the ILS-n configuration will have to be true for the matching ILS-$2n$ configuration scan chains as well. In the figure, we combine SC1 with SC2, and combine SC3 with SC4 for the ILS-n configuration. If SC1 was compatible with SC3 and SC2 was compatible with SC4, then it follows that SC12 would be compatible with SC34 for the ILS-$2n$ configuration.

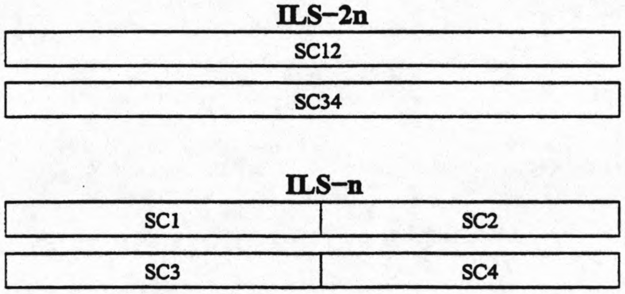


Figure 4.2 Example of Compatibility Relations between ILS Configurations

To take advantage of this fact, we implemented a simple algorithm where, following the regular procedure for Multiple Group ILS, an incompatibility graph for a small chain length configuration was generated first, such as ILS-16 for S38584.1. Then, an incompatibility graph for ILS-32 was generated. For every edge in this graph, we looked in the graph of ILS-16 and compared the 'extended' chains to see if it could be removed. Unfortunately, using this algorithm did not produce any better results for any of the tested circuits. However, this may be beneficial for larger circuits as there are more potential incompatibilities that could possibly be reduced.

4.2.2 Edge reduction using multiple ATPG runs

It is possible to reduce the number of edges in an incompatibility graph by performing different ATPG runs on the same circuit. The reason for this is that for an incompatibility graph produced for one ATPG run, the incompatibilities are not guaranteed between the scan chains, they are just specific for that particular set of test patterns produced by the ATPG run. Thus, if a different set of test patterns were produced for the same circuit by changing the ATPG in some manner, then certain chains could become compatible whereas in the previous run they were incompatible. The corresponding edge from the original incompatibility graph could then be removed.

The procedure for our implementation of this idea is shown in Figure 4.3. First, the regular procedure for producing an incompatibility graph for an ILS configuration was followed, and the graph was then colored. Then a new run was performed, in which the original netlist was altered so that were two groups. The first group contained the nodes from the largest group found from the coloring of the original graph, and the other group contained the nodes from all the other originally produced groups. The idea is that by constraining the ATPG so that one set of chains receive the same inputs, we will produce a different set of compatibilities for the same circuit.

An ATPG run was then performed to determine the new set of undetectable faults. Since there are more inputs than in in the original run using broadcast mode, there will

(1) Generate a test set B under Broadcast Mode. Perform static compaction on B. Identify the set of faults, U, that are undetectable.

(2) Regenerate the netlist for serial mode. Using a *nonfilling* ATPG, generate a partially specified test set S tagerting only the faults in U. Perform static compaction on S.

(3) Perform compatability analysis for all patterns in S to find the optimal group configuration. Store the original incompatability graph, G_{orig}

(4) Find the largest size group out of all the groups generated. Set that as Group 1. Join all other groups into Group 2.

(5) Regenerate the netlist using these two groups. Generate a test set, and find the new set of faults U_{new} that are undetectable.

(6) Regenerate the netlist for serial mode, except still constrain all the scan chains in group 1 to the same scan-in pin. Using a *non-filling* ATPG, generate a partially specified test set S_{new} tagerting only the faults in U_{new}. Perform static compaction on S_{new}.

(7) Perform compatability analysis for all patterns in S_{new} and generate a new incompatabilty graph, G_{new}

(8) For every edge in G_{orig}, if the edge is not found in G_{new}, then remove it. After all possible edges are removed, perform graph coloring and, if further elimination is desired, go to step 4.

Figure 4.3 Procedure for Edge Reduction Using Multiple ATPG Runs

be fewer undetectable faults. The netlist is then regenerated for serial mode, with the exception that the chains included in the first group will receive the same input. The nonfilling ATPG is used for the new set of undetectable faults on this new netlist, and a new test set will be generated, of which a new incompatibility graph will be formed. This graph will be different from the original graph, as different constraints were put on the ATPG in the new run. Then compatibilities are compared between the two graphs; if an edge in the original graph is not present in the new graph, then it is removed.

In our experiments, we found that while some edges were removed, there was not a significant enough improvement where fewer groups can be produced. Again, this may not be the case if used on larger circuits. Also, there are many other methods for producing a different set of test patterns for the same circuit which could prove to be beneficial for edge reduction. More complex procedures, such as used by [22] could be used to get more accurate compatibility information.

4.2.3 Edge reduction using vector elimination

One other option explored was the idea of removing particular vectors from compatibility analysis so that less edges would be produced. This is because, in some cases, a vector produced for a fault may introduce so many incompatibilities between chains that it causes several new groups to be produced. In this case, it is better in terms of test data volume just to use serial mode for this fault and eliminate it from compatibility analysis when producing groups. This method is most useful when using Multiple Group ILS with a single mode. In this case a secondary serial mode would have to be added for all the vectors that are ignored in groups mode.

Our implementation of this method was as follows: for each partially specified vector investigated for compatibility analysis, determine the total number of bit positions, or columns, within the scan chain that are conflicting. If a vector exceeds a certain percentage of conflicting bit positions compared to the total number of flops in the scan chain, then ignore it for compatibility analysis and proceed to the next vector. All ignored vectors are then included in a separate serial mode. Like the other methods, there were no improvements to the four circuits tested, because of their relatively small size. However, we were able to produce favorable results with a given set of partially specified vectors for an industrial circuit, whose characteristics are listed in Table 4.1

Table 4.1 Characteristics of Given Industrial Circuit

Circuit	Num Inputs	Num Outputs	Num Flip-Flops	Scan Chains	Length Longest Chain	Total Faults
Circuit A	1373	1011	19 924	105	286	1.14M

This circuit has 105 scan chains that are variable in length, with the maximum scan chain length being 286 flops wide. For this circuit, it was found that 1422 patterns were needed for full fault coverage for using serial mode alone. Next, broadcast mode was applied and then for all the undetectable faults a nonfilling ATPG was used to generate 209 serial patterns, which were then analyzed for compatibility between scan chains. After applying the coloring algorithm, we found that 23 groups were needed to test these faults.

However, because we were not given the circuit, we were unable to perform ATPG runs to determine the exact amount of vectors needed thereafter. We then make the assumption that, for using a single groups mode, the number of vectors needed will be approximately the same the number of vectors used for serial mode alone. In the preceding chapter, it was shown that the number of total vectors remained approximately the same for every configuration of a circuit, regardless of configuration or what modes are being used. By using this assumption, it can be estimated how much test data volume is needed with the amount of groups that are produced. Similarly, by using our proposed vector elimination procedure, it is also possible to estimate the test data volume for both groups mode and serial mode.

Table 4.2 Test Data Volume for Circuit A Using Vector Elimination

>% Conflicting Columns Ignored	Num Vectors Ignored	Num Groups	Group Bits (Mb)	Serial Bits (Mb)	Total Bits (Mb)	Red Factor
Full Serial	1422*	-	-	30.28	30.28	1.00
Broadcast	209*	1	2.36	4.45	6.81	4.45
0% (None)	0	23	11.31	-	11.31	2.68
60%	3	18	9.27	0.063	9.34	3.24
50%	7	15	8.05	0.149	8.20	3.69
40%	10	12	6.83	0.213	7.05	4.30
30%	20	9	5.61	0.426	6.04	5.02
20%	26	8	5.21	0.554	5.76	5.26
15%	32	6	4.39	0.682	5.07	5.97
10%	42	5	3.99	0.895	4.88	6.21

*indicates number of serial vectors

The results for this process are shown in Table 4.2. Here, it is assumed that 1422 vectors are needed for groups mode, equivalent to the number of vectors needed for serial mode alone. Also we assume that no reverse-order fault simulation has taken place and that the number of vectors ignored in groups mode is equivalent to the number of vectors needed for serial mode. The first column in this table represents at what percentage of conflicting columns compared to total flops with the scan chain that a vector is ignored. For example, in the fourth row of data, 3 out of the 209 partially specified vectors were ignored from compatibility analysis because within those vectors, 60 percent or greater of the 268 possible bit positions had conflicting values. Then, the rest of the 206 vectors were analyzed for compatibility analysis, upon which it was found that 18 groups were needed. Using the assumed number of group vectors mentioned previously, this will require 9.27 Mbits of tester data. The three ignored vectors will require 0.063 Mbits of tester data for serial mode. Then the total amount of data bits needed to be stored will be 9.34 Mbits which is a 2.68 times better improvement than using the conventional serial (full-scan) mode alone.

Table 4.2 shows that using single groups mode alone produces worse results than using the traditional ILS broadcast and serial modes. However, by ignoring particular vectors for compatibility analysis, there is a significant reduction in incompatibilities, which causes the number of groups produced to decrease. In fact, by ignoring just 42 out of the 209 vectors, the number of groups significantly drops from 23 to 5. This results in better test data volume reduction factors than using traditional ILS methods. Thus, even though there are some assumptions made for this case, it is evident that using vector elimination for large circuits can be useful for test data volume reduction.

4.3 Summary

In this chapter, we presented various methods that could be used with the procedure described for Multiple Group ILS. All methods were implemented in order to reduce the number of groups that are needed to test all undetectable faults in broadcast mode, and thus reduce the test data volume required. Also it was explained that as circuits become larger, these methods have the potential to become more beneficial.

CHAPTER 5

ILS WITH TRANSITION FAULTS

In Chapter 2, it was shown that using an ILS architecture was effective in reducing both test data volume and test application time. However, testing was performed only for stuck-at faults. In this chapter, we show that the ILS methodology can be applied effectively for transition faults as well.

5.1 Preliminaries on Transition Faults

The transition fault model is used to detect local delay faults on a particular line in a circuit. There are two types of transition faults, *slow-to-rise* faults and *slow-to-fall* faults. For a slow-to-rise fault on a particular line, the signal on that line is initially 0 during the first clock cycle. During the next clock cycle, this line should transition to a 1. If the line is faulty, it will be unable to change in time, and will be read as a 0. To detect these types of faults, two vectors are needed [27]. The first vector initializes a value on the line, which in this case would be 0. The second vector excites the fault by putting the opposite value on the line, which in this case would be a 1, and then propagates the effect of the fault to a primary output. In fact, generation of the second pattern is equivalent to testing for a stuck at fault on that particular line. Note that for a slow-to-fall fault, the analysis is identical except all ones are replaced with zeros, and vice versa.

5.2 Transition Fault Implementation for Full-Scan Circuits

We modified our combinational ATPG, called ATOM [19], to produce two vector tests for transition faults. In order to use this combinational ATPG, a full-scan version of a circuit is required. However, for our implementation of transition faults, the functional justification technique was used, which requires the use two sequential time frames: one frame for the initialization of the line, where all scan elements were fully controllable, and a second frame for testing of the line, where scan elements should contain the values obtained from the first time frame. In order to handle this, we used the technique described in [15]. This technique, which is defined here as the two-frame technique, duplicates a given sequential circuit into two 'frames,' as shown in Figure 5.1.

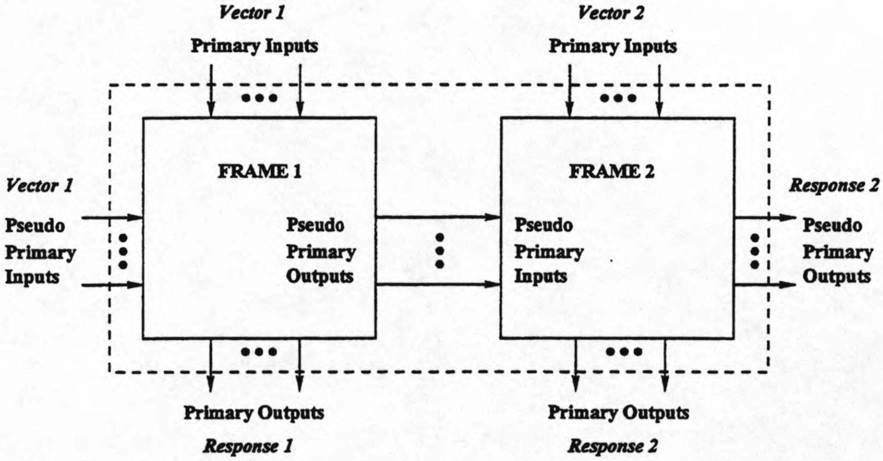


Figure 5.1 Two Frame Circuit Using the Functional Justification Technique [15]

Here the outputs of the scan elements in the first frame are treated as primary inputs, the inputs of the scan elements in the first frame are connected to outputs of the scan elements of the second frame, and finally the inputs of the scan elements in the second frame are treated as primary outputs. By modifying the circuit in this way, we can fully control the scan elements in the first frame, while the values of the scan elements in the second frame are dependent on the values of the scan elements in the first frame. In addition, both frames together form a combinational circuit, allowing the use of our modified combinational ATPG. Test generation can be then be achieved by initializing a particular line in the first frame to a value, while testing the corresponding line in the second frame for the appropriate stuck-at fault.

As shown in Figure 5.1, two vectors are generated. The first vector will include the pseudo primary inputs, which will be shifted into the scan chain. Then the primary input from the first vector will be applied in one clock cycle. In the next clock cycle, the primary inputs from the second vector will be applied, and the output response, denoted as *Response 2* of the circuit will be monitored to detect the fault. This will consist of the pseudo-primary outputs which will be shifted out from the scan chains, and also the primary outputs, which are directly observed.

5.2.1 Modifications to the ATPG and fault simulator

Using the two-frame technique, only minor modifications were needed to convert our stuck-at fault ATPG into one that tested for transition faults. Figure 5.2 illustrates our modification to the ATPG for a slow-to-rise fault on the line in between points A and

B. The idea is to follow the same procedure for testing a stuck-at-0 fault in the second frame, except now we add an AND gate and an inverter in between corresponding lines between the first and second frame, which forces them to be opposite in value in order to propagate the fault forwards. In the figure, the value of the line in the first frame must be 0 in order for the fault to propagate out on the second frame. Note that for a slow-to-fall fault, an OR gate is added instead of an AND gate.

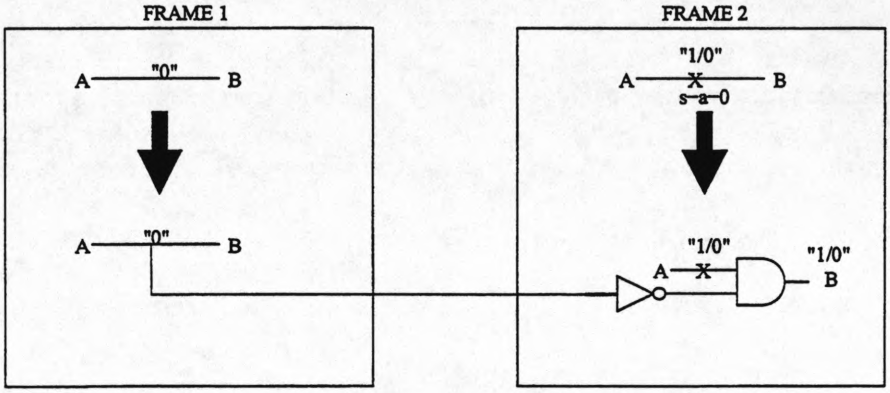


Figure 5.2 Modification of the ATPG to Inject a Transition Fault

For modification of our fault simulator, the same procedure for fault detection of stuck-at faults is followed, except we also check that the corresponding line in the first frame is of opposite value.

5.3 Test Generation Procedure for ILS

The two frame technique can be applied for ILS broadcast mode as well. This is accomplished using the same process as two-frame technique for full scan circuits, except all scan elements that were previously treated as primary inputs are now connected to an appropriate scan-in pin, depending on what order the scan elements are arranged. For example, for a scan chain of length 100, an ILS-25 configuration would have 4 scan chains, where scan elements 1, 26, 51, and 76 would be positioned so that they receive identical scan-in data. In addition to converting the circuit, a table is generated cross-referencing nodes in the first frame with the corresponding nodes in the second frame. This table is then used by the test generator and fault simulator.

The test generation procedure is shown in Figure 5.3. First the netlist is converted an ILS configuration as previously described. After that step, the procedure is identical to that of the one presented in Chapter 2, except here we are testing faults only in the second frame. Using our modified ATPG, this will automatically justify the first frame for the conditions needed for a transition fault. The collapsing of transition faults in step 2 was based on the transition fault equivalence relation described in [27].

(1) Convert netlist for use with ILS broadcast mode using the two-frame technique, and generate cross-reference table.

(2) Generate a test set B under Broadcast Mode for all collapsed faults in frame 2. Perform static compaction on B. Identify the set of faults, U, that are undetectable.

(3) Generate a test set S under Serial Mode targeting only the faults in U. Perform static compaction on S.

(4) Fault simulate test set S for all faults, and remove the detected faults from the complete fault list.

(5) Perform static compaction on test set B using the remaining faults from the previous step or regenerate B under Broadcast Mode using only these faults.

(6) Output the final test set T = B ∪ S.

Figure 5.3 Test Generation Procedure for ILS for Transition Faults

5.4 Experimental Results

The test generation procedure was performed using the same four circuits tested in Chapter 2. First test generation on the circuit using the two-frame technique for full scan circuits was performed for baseline comparison purposes. All runs were performed on the same machine used in Chapter 2, an AMD 900 MHz using 256 MB of memory. All tests were generated using our modified versions of ATOM and the fault simulator, and static compaction was performed using a simple single detection forward and reverse pass technique [1].

Before defining the calculation of test data volume and test application time for transition faults, some clarification is needed concerning test patterns. Using two-frame technique, the ATPG generates a test pattern that includes both test vectors needed to test a transition fault. Therefore, for simplicity, we will refer to a single test pattern as containing both the first vector, which includes primary and pseudo inputs, and the second vector, which includes just primary inputs. Using this definition, and assuming the parallel access technique is used, the calculation for test application time is the same as in Chapter 2, since for each test pattern a vector needs to be shifted in and a response needs to be shifted out. The equation for test application time, measured by the number of test cycles, is computed as

$$F + (1 + F) * P$$

where F is the number of flip-flops and P is the number of test patterns in the test set. Since a test pattern contains both vectors needed for a transition fault, the test data volume is calculated as

$$(2 * PI + F) * P$$

where PI is the number of primary inputs, F is the number of flip-flops, and P is the number of test patterns.

Table 5.1 Results for Full Scan Circuit for Transtition Faults

Circuit	Total Faults	Detected Faults	Test Vectors	Test Cycles	Test Data (bits)	Total Time (secs)
s13207.1	15 602	13 612	973	622 385	741 426	117
s15850.1	19 046	15 970	855	457 959	588 240	162
s38417	49 738	48 753	2183	3 575 207	3 693 636	462
s38584.1	61 254	56 376	1973	2,816,897	2 963 446	590

Table 5.1 shows the results from test generation of the full-scan circuits for transition faults followed by static compaction. The second and third columns show that there is a significantly greater number of transition faults to test compared to stuck-at faults. This is because there are fewer fault equivalence rules for transition faults than stuck-at faults, which causes less faults to be collapsed. This leads to many more vectors (column 4) being generated, which will lead to much greater test cycles (column 5) and test data volume (column 6) than those for stuck-at faults. Since a two-frame circuit is being used, there are twice as many gates that the test generator and fault simulator have to traverse

through. Combined with the fact that there are many more faults to test, this leads to much longer run times than those for stuck-at faults, as shown by column 7.

Next, test generation is performed for the circuits using the ILS methodology, using the method described in the previous section. Various configurations for each circuit were tested in order to find the configuration that produced the best results. The longest runtime was 2 hr and 55 min for the 8 configurations of circuit s38584. As mentioned previously, because of the increased amount of gates and faults needed for testing, this leads to much longer runtimes than those for stuck-at faults.

Computations for test application time and test data volume in broadcast mode are similar to the ones described in Chapter 2, except for our new definition of a test pattern. The equation for a test application time for broadcast mode is computed as

$$F_{LSC} + (1 + F_{LSC}) * P_B$$

where F_{LSC} is the length of the longest segment and P_B is the number of broadcast patterns required. The test data volume for broadcast mode is computed as

$$(2 * PI + F_{LSC}) * P_B$$

where PI is the number of primary inputs, F_{LSC} is the length of the longest chain in Broadcast Mode, and P_B is the number of broadcast patterns required.

Table 5.2 shows the experimental results for the test application time required for various ILS configurations of all tested circuits. The columns in these table represent the circuit being tested, the ILS configuration, the number of serial patterns required, the tester cycles required for these serial patterns, the number of scan chain segments in broadcast mode, the final number of broadcast patterns needed after RFS, the tester cycles required to apply these patterns, the total tester cycles required including serial and broadcast mode, and finally the test application time reduction factor. The reduction factor is calculated as the ratio of the total cycles needed for the conventional full scan circuit (indicated by 'Baseline') to the total cycles needed for a particular ILS configuration.

The experimental results for test data volume for various ILS configurations of the tested circuits are shown in Table 5.3. The columns in this table represent the circuit being tested, the ILS configuration, the number of serial patterns required, the amount of data bits that needs to be stored in the tester for these serial patterns, the number of scan chain segments in broadcast mode, the final number of broadcast patterns needed after RFS, the amount of data bits that needs to be stored in the tester for these broadcast patterns, the total amount of data bits that needs to be stored for both broadcast and serial modes, and finally the reduction factor. The reduction factor is calculated as the ratio of the total bits needed to be stored for the full scan circuit to the total bits needed to be stored for a particular ILS configuration.

Table 5.2 Test Application Time Reduction Using ILS for Transition Faults

		Serial Mode		Broadcast Mode				
Circuit	**Config**	**Patterns**	**Test Cycles**	**Num Chains**	**Patterns**	**Test Cycles**	**Total Cycles**	**Reduction Factor**
s13207.1	Baseline	973	622 385	-	0	0	622 385	1.00
	ILS-360	18	12 140	2	970	350 530	362 670	1.72
	ILS-180	34	22 364	4	959	173 759	196 123	3.17
	ILS-100	156	100 322	7	833	84 233	184 555	3.37
	ILS-90	294	188 504	8	688	62 698	251 202	2.48
	ILS-50	279	178 919	13	684	34 934	213 853	2.91
	ILS-45	351	224 927	15	618	28 473	253 400	2.46
	ILS-25	414	265 184	26	532	13 857	279 041	2.23
s15850.1	Baseline	855	457 959	-	0	0	457 959	1.00
	ILS-300	8	4814	2	850	256 150	260 964	1.75
	ILS-150	102	55 104	4	756	114 306	169 410	2.70
	ILS-100	130	70 084	6	728	73 628	143 712	3.19
	ILS-75	152	81 854	8	712	54 187	136 041	3.37
	ILS-50	198	106 464	11	655	33 455	139 919	3.27
	ILS-25	274	147 124	22	590	15 365	162 489	2.82
s38417	Baseline	2183	3 575 207	-	0	0	3 575 207	1.00
	ILS-1024	0	1636	2	2138	2 192 474	2 194 110	1.63
	ILS-512	25	42 561	4	2155	1 106 027	1 148 588	3.11
	ILS-256	33	55 657	7	2118	544 582	600 239	5.96
	ILS-128	71	117 863	13	2071	267 287	385 150	9.28
	ILS-64	270	443 626	26	1856	120 704	564 330	6.34
	ILS-32	612	1 003 480	52	1553	51 281	1 054 761	3.39
	ILS-16	1114	1 825 254	103	982	16 710	1 841 964	1.94
s38584.1	Baseline	1973	2 816 897	-	0	0	2 816 897	1.00
	ILS-800	24	35 674	2	1900	1 522 700	1 558 374	1.81
	ILS-400	66	95 608	4	1885	756 285	851 893	3.31
	ILS-200	169	242 589	8	1819	365 819	608 408	4.63
	ILS-128	228	326 782	12	1742	224 846	551 628	5.11
	ILS-100	321	459 493	15	1665	168 265	627 758	4.49
	ILS-64	491	702 083	23	1437	93 469	795 552	3.54
	ILS-32	671	958 943	45	1269	41 909	1 000 852	2.81
	ILS-16	993	1 418 437	90	946	16 098	1 434 535	1.96

Table 5.3 Test Data Volume Reduction Using ILS for Transition Faults

		Serial Mode		Broadcast Mode				
Circuit	**Config**	**Patterns**	**Mem bits**	**Num Chains**	**Patterns**	**Mem bits**	**Total bits**	**Reduction Factor**
s13207.1	Baseline	973	741 426	-	0	0	741 426	1.00
	ILS-360	18	13 716	2	970	469 480	483 196	1.53
	ILS-180	34	25 908	4	959	291 536	317 444	2.34
	ILS-100	156	118 872	7	833	186 592	305 464	2.43
	ILS-90	294	224 028	8	688	147 232	371 260	2.00
	ILS-50	279	212 598	13	684	119 016	331 614	2.24
	ILS-45	351	267 462	15	618	104 442	371 904	1.99
	ILS-25	414	315 468	26	532	79 268	394 736	1.88
s15850.1	Baseline	855	588 240	-	0	0	588 240	1.00
	ILS-300	8	5504	2	850	385 900	391 404	1.50
	ILS-150	102	70 176	4	756	229 824	300 000	1.96
	ILS-100	130	89 440	6	728	184 912	274 352	2.14
	ILS-75	152	104 576	8	712	163 048	267 624	2.20
	ILS-50	198	136 224	11	655	133 620	269 844	2.18
	ILS-25	274	188 512	22	590	105 610	294 122	2.00
s38417	Baseline	2183	3 693 696	-	0	0	3 693 696	1.00
	ILS-1024	0	0	2	2138	2 309 040	2 309 040	1.60
	ILS-512	25	42 300	4	2155	1 224 040	1 266 340	2.92
	ILS-256	33	55 836	7	2118	66 0816	716 652	5.15
	ILS-128	71	120 132	13	2071	381 064	501 196	7.37
	ILS-64	270	456 840	26	1856	222 720	679 560	5.44
	ILS-32	612	1 035 504	52	1553	136 664	1 172 168	3.15
	ILS-16	1114	1 884 888	103	982	70 704	1 955 592	1.89
s38584.1	Baseline	1973	2 963 446	-	0	0	2 963 446	1.00
	ILS-800	24	36 048	2	1900	1 664 400	1 700 448	1.74
	ILS-400	66	99 132	4	1885	897 260	996 392	2.97
	ILS-200	169	253 838	8	1819	502 044	755 882	3.92
	ILS-128	228	342 456	12	1742	355 368	697 824	4.25
	ILS-100	321	482 142	15	1665	29 3040	775 182	3.82
	ILS-64	491	737 482	23	1437	201 180	938 662	3.16
	ILS-32	671	1 007 842	45	1269	137 052	1 144 894	2.59
	ILS-16	993	1 491 486	90	946	87 032	1 578 518	1.88

The results from both Tables 5.2 and 5.3 show the same trends as the tables for test application and test data volume reduction explained in Chapter 2. The maximum reduction factors for both test data volume and test application time for all circuits remain about the same as those for stuck-at faults, which leads to the conclusion that using ILS for transition faults is just as effective in reducing test data volume and test application time as stuck-at faults. Note that the configurations where the maximum reduction factors are found are close or identical to those for stuck-at faults.

5.5 Summary

In this chapter, we extended the ILS algorithm for use with transition faults using the functional justification technique. By using the two-frame technique, we were able to make minor modifications to the ATPG and fault simulator in order to produce the two-vector test set required for a transition fault. Although runtimes were significantly longer, it was shown that using the ILS methodology for transition faults was equally effective in reducing test data volume and test application time as using ILS for stuck-at faults.

CHAPTER 6

CONCLUSIONS AND FUTURE RESEARCH

In this thesis, we presented an analysis of the Illinois Scan Architecture (ILS). It was shown that by using this methodology, a significant reduction in both test application and test data volume were achieved. This will thereby reduce the testing cost involved, which is a major concern in the overall manufacturing cost of ICs.

It was shown that as the length of the scan chain segments for an ILS configuration became shorter, the cost of using serial patterns became more expensive. Therefore, we proposed an algorithm, called Multiple Group ILS, that would replace the use of serial mode with a less expensive mode, called groups mode, obtained from the compatibility analysis of scan chains. It was shown that using both broadcast and groups modes, or even just a single groups mode, produces a much more significant reduction in test data volume and test application time than using broadcast and serial modes.

We proposed several variations to the algorithm for compatibility analysis of scan chains that may help in reducing the number of groups produced for groups mode. We showed that by using vector elimination along with compatibility analysis, we could use groups mode and serial mode to produce better results than using broadcast and serial modes, as well as groups mode alone.

Finally, we showed that the ILS methodology could be applied for use with transition faults as well as stuck-at faults. Using the two-frame technique for functional justification, we were able to use a combinational ATPG and easily modify it to produce test patterns to detect transition faults. It was shown that although runtimes were greatly increased, using ILS produced similar reduction in test data volume and test application for transition faults as it had for stuck-at faults.

6.1 Future Research

All experiments performed in this thesis were performed on circuits, which by today's standards, are relatively small. More useful results could be obtained if tests were performed on larger industrial circuits. This is especially important for Multiple Group ILS, as the benefit of using groups increases with larger numbers of scan chains. Also with larger incompatibility graphs, the proposed techniques of group reduction would become more useful.

In our algorithm for Multiple Group ILS, compatibility analysis was performed using an incompatibility graph. If a compatibility graph was used instead, better results may be obtained. In addition, several algorithms for the analysis of compatibility graphs [11, 12, 22, 28] used in BIST have been proposed that also include the use of inverse nodes in their analysis. Future work in this area could be to adapt these algorithms for use with compatibility graphs for scan chains.

Experiments performed on using ILS with transition faults used only a simple compaction technique. The number of test patterns required could be decreased if better compaction techniques were used. Furthermore, the incremental algorithm proposed in [7] could be used for transition faults to reduce runtimes. Finally, the Multiple Group ILS algorithm could be extended for transition faults to further reduce test data volume and test application time.

REFERENCES

[1] V. Agrawal and M. Bushnell, *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits.* Norwell: Kluwer Academic Publishers, 2000.

[2] F. F. Hsu, K. M. Butler, and J. H. Patel, "A case study on the implementation of the Illinois scan architecture," in *Proc. of the Int. Test Conf.*, October 2001, pp. 538-547.

[3] D. Das and N. A. Touba, "Reducing test data volume using external/LBIST hybrid test patterns," in *Proc. Int. Test Conf.*, October 2000, pp. 115-122.

[4] G. Kiefer, H. Vranken, E. J. Marinissen, and H-J. Wunderlich, "Applications of deterministic logic BIST on industrial circuits," in *Proc. Int. Test Conf.*, October 2000, pp. 105-114.

[5] A. Jas, B. Pouya, and N. A. Touba, "Virtual scan chains: A means for reducing scan length in cores," in *Proc. of the IEEE VLSI Test Symp.*, April 2000, pp. 73-78.

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

[7] A. R. Pandey and J. H. Patel, "An incremental algorithm for test generation in Illinois scan architecture based designs," in *Proc. Design, Automation, and Test in Europe*, March 2002, pp. 368-375.

[8] A. R. Pandey and J. H. Patel, "Reconfiguration technique for reducing test time and test data volume in Illinois scan architecture based designs," in *Proc. IEEE VLSI Test Symp.*, April 2002, pp. 9-15.

[9] A. R. Pandey, "Test time and test data volume reduction techniques for VLSI circuits," M.S. thesis, University of Illinois at Urbana-Champaign, 2001.

[10] S. Samaranayake, E. Gizardski, N. Sitchinava, F. Neuveux, R. Kapur, and T. W. Williams, "A reconfigurable shared scan-in architecture," in *Proc. IEEE VLSI Test Symp.*, April 2003, pp. 9-14.

[11] C. Chen and S. K. Gupta, "A methodology to design efficient BIST test pattern generators," in *Proc. of the Int. Test Conf.*, October 1995, pp. 814-823.

[12] K. Chakrabarty, B. Murray, J. Liu, and M. Zhu, "Test width compression for built-in self testing," in *Proc. of the Int. Test Conf.*, October 1997, pp. 328-337.

[13] O. Bula, J. Moser, J. Trinko, M. Weissman, and F. Woytowich, "Gross delay defect evaluation for a CMOS logic design system product," *IBM Journal of Research and Development*, volume 34, no. 2/3, pp. 325-328, March/May 1990.

[14] P. Franco, S. C. Ma, Y-C.Chang, R. Stokes, W. Farwell, and E. J. McCluskey, "Analysis and detection of timing failures in an experimental test chip," in *Proc. of the Int. Test Conf.*, October 1996, pp. 691-700.

[15] I. Hamzaoglu and J. H. Patel, "Compact two-pattern test set generation for combinational and full scan circuits," in *Proc. of the Int. Test Conf.*, October 1998, pp. 944-953.

[16] L. N. Reddy, I. Pomeranz, and S. M. Reddy, "COMPACTEST-II: A method to generate compact two-pattern test sets for combinational logic circuits," in *Proc. Int. Conf. on Computer-Aided Design*, November 1992, pp. 568-574.

[17] J. Savir, "Skewed-load transition test: part I, calculus," in *Proc. Int. Test Conf.*, October 1992, pp. 705-713.

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

[19] I. Hamzaoglu and J. H. Patel, "New techniques for deterministic test pattern generation," in *Proc. IEEE VLSI Test Symp.*, April 1998, pp. 446-452.

[20] X. Lin, J. Rajski, I. Pomeranz, and S. M. Reddy, "On static test compaction and test pattern ordering for scan designs," in *Proc. Int. Test Conf.*, October 2001, pp. 1088-1097.

[21] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *Proc. Int. Symp. on Circuits and Systems*, May 1989, pp. 1929-1934.

[22] I. Hamzaoglu and J. Patel, "Reducing test application time for built-in-self-test pattern generators," in *Proc. of the IEEE VLSI Test Symp.*, April 2000, pp. 969-975.

[23] D. Brelaz, "New methods to color the vertices of a graph," *Communications of the ACM*, vol. 22, no. 4, pp. 251-256, April 1979.

[24] E. C. Sewell, "An improved algorithm for exact graph coloring," in *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 26, 1996, pp. 359-373.

[25] A. Mehrotra and M. A. Trick, "A column generation approach for graph coloring," *Journal on Computing*, vol. 8, no. 4, pp. 344-354, Fall 1996.

[26] W. Klotz, "Graph coloring algorithms," Mathematics Report, Technical University Clausthal, May 2002, pp. 1-9.

[27] J. A. Waicukauski, E. Lindbloom, B. K. Rosen, and V. S. Iyengar, "Transition fault simulation," *IEEE Design and Test of Computers*, vol. 4, pp. 32-38, April 1987.

[28] E. Gizdarski and H. Fujiwara, "Fault set partition for efficient width compression," in *Proc. Asian Test Symp.*, November 2002, pp. 194-199.