The Thesis Committee for Kangjoo Lee
Certifies that this is the approved version of the following Thesis:

Designing An Efficient Test Pattern Generator Using Input Reduction with Linear Operations

APPROVED BY
SUPERVISING COMMITTEE:

Nur A. Touba, Supervisor

Nan Sun

Designing An Efficient Test Pattern Generator Using Input Reduction with Linear Operations

by

Kangjoo Lee

Thesis

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Engineering

The University of Texas at Austin

May 2018

Acknowledgements

It is indeed a great honor to have Prof. Nur A. Touba as my supervising advisor throughout my graduate program at the University of Texas at Austin. Without his consistent support and sincere advice, I doubt that I could accomplish all my works up to this far. Prof. Touba is more than just academic supervisor to me. He ardently supports not only to accomplish my Thesis successfully but also to guide me in the right direction whenever I faced in an obstacle in my life. I would also like to thank Prof. Nan Sun for being as a reader and reviewing my work.

Abstract

Designing An Efficient Test Pattern Generator Using Input Reduction with Linear Operations

Kangjoo Lee, MSE

The University of Texas at Austin, 2018

Supervisor: Nur A. Touba

Advances in fabrication technology have resulted in more complicated systems, being used in ever increasing numbers of applications. The large increase in transistor counts versus the number of pins on the chip has made VLSI testing much harder than ever before. Denser integrated circuits chips increase the required test cases enormously for comprehensive testing of a chip. This results in expensive test cost and long test time. In this thesis, an improved method for on-chip test pattern generation is proposed. It generates a complete test set more efficiently by using input reduction with linear operations. Input reduction for pseudo-exhaustive test pattern generation based on compatible and inverse-compatible relationships between inputs has been proposed in the past. This work extends the concept by using linear combinations of inputs to generate other inputs as a means for further input reduction. Results are presented showing the improvements that can be obtained.

Table of Contents

List of Tables viii

List of Figures ix

Chapter 1: Introdcution 01

Chapter 2: Background Knowledge on VLSI Testing 02

Chapter 3: Pseudo-Exhaustive Testing Using a Compatibility Matrix 03

3.1 Introdcution for Pseudo-Exhaustive Testing 03

3.2 Fundamental Concept of Generating Compatibility Matrix 05

3.3 Algorithm For Generating Pesudo-Exhaustive Test Set by Input Redcution 07

3.4 Experimental Results 08

3.5 Drawback 10

Chapter 4: Input Reduction Technique Based on Linear Operations 11

4.1 Idea of Using Linear Operations in Pseudo-Exhaustive Testing 11

4.2 Using Gauss-Jordan Elimination to Find Basis 11

4.3 Algorithm For Input Reduction Using Linear Operations 13

4.4 Experimental Results 14

4.5 Drawback 15

Chapter 5: Combined Method 16

5.1 Introdcution for Combined Method 16

5.2 Combined Algorithm 16

5.3 Experimental Results 16

Chapter 6: Conclusion.....19

Bibliography 20

Vita.....21

List of Tables

Table 1: If two primary inputs are compatible relation.....08

Table 2: If two primary inputs are inversely compatible relation.....08

Table 3: Using compatilibiy matrix for test sets containing average number of logic 'X' 09

Table 4: Using compatilbility matrix for test sets containing a lot of logic 'X' 10

Table 5: Using Gauss-Jordan Elimination for test sets containg average number of logic 'X'.....15

Table 6: Using Gauss-Jordan Elimniation for test sets containg a lot of logic 'X'.....15

Table 7: Using combined method for test sets containg average number of logic 'X' 17

Table 8: Using combined method for test sets containg a lot of logic 'X'.....17

List of Figures

Figure 1: Benchmark circuit for C17 03

Figure 2: Depdence Matrix for benchmark circuit C17 04

Figure 3: Test Pattern Generator design with inverted interconnections for C17 05

Figure 4: Compatibility Matrix for C17 06

Figure 5: Forming Basis using Gauss-Jordan Elimination 12

Figure 6: Pseudo-exhaustive Test Pattern Generator design using compatibility matrix 13

Figure 7: Pseudo-exhaustive Test Pattern Generator design using linear operations 13

Figure 8: Graphical representation of compression rate with average number of logic 'X' 18

Figure 9: Graphical representation of compression rate with a lot of logic 'X' 18

Chapter 1: Introduction

Advancement in fabrication technology has allowed more transistors in smaller area thereby increasing the complexity and density in very-large-scale integration (VLSI) circuits. The semiconductor industry has boomed for several decades with the world-wide market increasing rapidly every year. One of the challenges is keeping down test costs as the number of tests required and the number of transistors per pin has greatly increased. The time required to bring in all test data from an external tester through the chip pins has been going up thereby increasing test costs. Better and more efficient ways to test complex and dense integrated circuits are needed.

In this thesis, we focus on a method to reduce test cost by efficiently designing an on-chip test pattern generator that avoids the need to bring data from an external tester. This helps to reduce test time as well as has the capability to perform tests in the field when no tester is available. A new method for designing an on-chip test pattern generator using an improved method for input reduction that considers linear combinations of inputs to generate other inputs is proposed.

The thesis is organized as follows. In Chapter 2, background information on VLSI testing is presented. In Chapter 3, the conventional method for performing input reduction based on using a compatibility matrix is described. In Chapter 4, the proposed methodology for using Gauss-Jordan Elimination to identify linear combinations of inputs to improve input reduction is presented. Each of those two techniques, *compatibility matrix* and *linear combinations*, has its own distinct strength for a given test set. However, they also have some drawbacks for certain circumstances as well. Fortunately, the two design techniques can be used together to complement each other, so in Chapter 5, our final design implementation using both techniques is presented. Chapter 6 concludes the thesis.

Chapter 2: Background Knowledge on VLSI Testing

The fundamental procedure of VLSI testing is performed within three main stages. The first stage is where we need to generate a test sequence or input test stimulus to apply for a circuit under test (CUT). The second stage obtains the resulting values for every test sequences applied. Lastly, there is a detector which compares the resulting values with fault-free cases and determines whether either the CUT passes or fails [1-3].

Therefore, we need to come with better and more efficient methods for the first two stages to obtain the fundamental goal of VLSI testing, which is producing high test quality while having low test cost. We also define the first stage as test generation and the second stage as design for testability (DFT). Test generation refers to a definition of developing or generating an efficient way to come with a compact and small test stimulus for a corresponding CUT while improving test quality. DFT is a method of modifying a circuit design to be easier and simpler in respect to the test scenario so that we can ultimately reduce total number of test cases and test time for analyzing the circuit design [4].

There are numerous different design techniques for test generation and design for testability. Test generation is mostly computed by an automatic test pattern generation (ATPG) method for VLSI testing. There are five well-known algorithms for ATPG, which are the D-Algorithm, PODEM, FAN, TOPS, and Socrates [5]. Design for testability techniques are mainly categorized into Ad hoc DFT, scan design, or built-in self-test (BIST) [5]. In this thesis, we are going to focus on developing an efficient built-in self-test DFT scheme which is based on applying a given test set in a reasonable amount of time. It is based on designing an efficient test pattern generator by using input reduction with linear operations.

Chapter 3: Pseudo-Exhaustive Testing Using a Compatibility Matrix

3.1 Introduction For Pseudo-Exhaustive Testing

The goal of designing an efficient test pattern generator (TPG) is to generate a test set that provides complete fault coverage with a reasonable test length. If we are applying an exhaustive test set to a circuit under test, this requires 2^n test vectors where n is the total number of primary inputs. An exhaustive test set is guaranteed to provide the highest percentage of fault coverage for each circuit design. However, test cost and test time increase dramatically with increasing n because the number of test vectors is proportional to 2^n. For this reason, using an exhaustive test set is considered as an impractical testing methodology for most VLSI testing applications. An alternative is to use an appropriately generating pseudo-exhaustive test set which can reduce the required test vectors enormously while still providing same percentage of fault coverage as an exhaustive test set does.

We will use benchmark circuit C17, which is the simplest one from ISCAS85, to provide visual explanation and more comprehensive knowledge on pseudo-exhaustive testing.

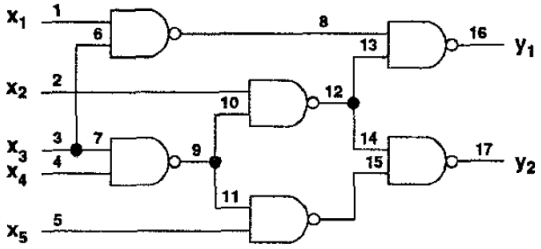

Figure 1: Benchmark circuit for C17 [8]

Based on the Figure 1, if we simply apply an exhaustive test set for C17, it will require a total number of $2^5 = 32$ test vectors because there are 5 primary inputs. We can generate this with a 5 bit *linear-feedback shift register (LFSR)* circuit. However, C17 is a partial dependence circuit (PDC), which means that there are some primary outputs which do not depend on all primary inputs. Therefore, we can generate a dependence matrix that

defines the direct relationship between primary inputs and primary outputs. The basic rule for generating dependence matrix is given by

$$D_{I,J} = 1 \; iff \; Output \; i \; depends \; on \; input \; j \quad [6]$$

and the corresponding dependence matrix for C17 is shown in Figure 2-a.

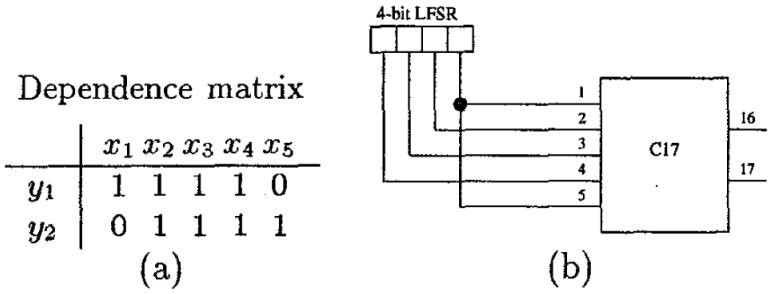

Figure 2: Dependence Matrix for benchmark circuit C17 [8]

Based on the dependence matrix, we notice that primary input x_1 is only used at y_1 and primary input x_5 is only used at y_2. Therefore, we can combine those two bits into one test signal while constructing the LFSR for C17. As a result, we can instead use a 4-bit LFSR to test C17 by defining the dependence matrix for this circuit and the following result is shown in Figure 2-b [6].

There are even more efficient designs to reduce the test set further, as was proposed by Chen and Gupta in [8]. The idea in their paper was to start with a test set that detects all faults in a circuit, and then identify compatible or inverse compatible inputs from that test set. Using their design technique, the number of inputs can be reduced all the way down to a 2-bit LFSR as illustrated in Fig. 3. The test set is shown in Fig. 3(a) where each row is a test vector and each column is an input. As can be seen, column 3 and column 5 are identical, so x_3 and x_5 can be driven by the same test signal. And columns 2 and 4 are equal to column 1 inverted, so x_2 and x_4 can be driven by the complement of the test signal used for x_1. The final solution is shown in Figure 3-b.

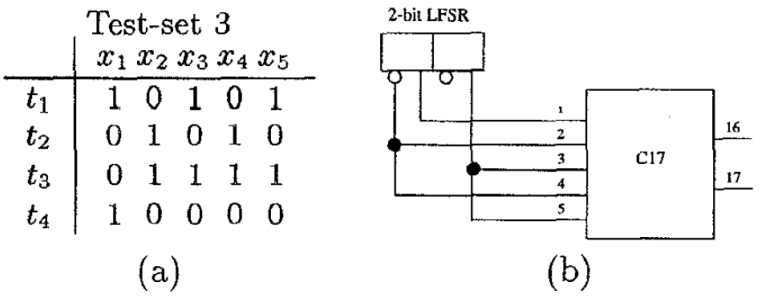

Figure 3: Test Pattern Generator design with inverted interconnections for C17 [8]

3.2 FUNDAMENTAL CONCEPT OF GENERATING COMPATIBILITY MATRIX

As illustrated in Section 3.1, we would like to obtain a pseudo-exhaustive test case from an appropriate set of test vectors to test a circuit comprehensively. The goal is to minimize the total size of the test pattern generator as much as possible which helps to reduce the test length as well as reduce the area overhead. A detailed explanation of obtaining a complete test set, which refers to a test set that is guaranteed to have an optimal size of test set for detecting all single stuck-at fault in a circuit, is explained in paper [7]. In this paper, we will assume a complete test set is available and use that as our starting point for designing an efficient test pattern generator.

The conventional pseudo exhaustive testing with input reduction technique that was originally proposed in [7] identifies compatible and inverse compatible relationships among inputs in the test set. This was illustrated in Section 3.1. Based on the test set for C17, we confirmed that primary inputs x_3 and x_5 can be merged into one test signal because they are exactly equivalent each other for all their test vectors from t_1 to t_4. Within this following condition, we define x_3 and x_5 as compatible inputs and only required to have one test signal in the test pattern generator for testing those two primary inputs.

Definition 1 (Compatible Inputs) [8-9] – Two inputs x_i and x_j that can be shorted together without introducing any redundant stuck-at fault in the circuit are said to be compatible.

We also confirmed that primary inputs x_1 and x_2 can be merged into one test signal since they are exactly inverted from each other for all test vectors from t_1 to t_4. Within this given condition, we define x_1 and x_2 are inversely compatible inputs each other.

Definition 2 (Inversely Compatible Inputs) [8-9] – Two inputs that can be shorted via an inverter without introducing any redundant stuck-at fault in the circuit are said to be inversely compatible.

If primary inputs are neither of those two given definitions, we call them unrelated to each other or incompatible to each other.

Definition 3 (Unrelated Inputs) [8-9] – Two inputs that cannot be shorted without introducing any redundant stuck-at fault in the circuit are said to be unrelated inputs.

Based on the definitions 1 to 3, we can generate a *compatibility matrix* for any complete test set and a detail explanation for deriving a compatibility matrix is explained in Definition 4. The compatibility matrix for C17 is shown in Figure 4.

Definition 4 (Compatibility Matrix) [8-9] – The compatibility matrix for an *n*-input CUT is an *n* x *n* matrix. The upper triangular matrix represents the compatibility relations, while the lower triangular matrix represents the inverse compatibility relations. The entry (i, j) is 1 (0) if the inputs x_i and x_j are (inversely) compatible. A "-" in the entry (i, j) indicates that the input x_i and x_j are incompatible. The compatibility matrix contains a total of $2\binom{n}{2} = n^2 - n$ entries.

	x_1	x_2	x_3	x_4	x_5
x_1	X	-	-	1	1
x_2	0	X	1	1	-
x_3	-	-	X	-	1
x_4	0	-	-	X	1
x_5	0	-	-	-	X

Figure 4: Compatibility Matrix for C17 [8]

3.3 ALGORITHM FOR GENERATING PSEUDO-EXHAUSTIVE TEST SET BY INPUT REDUCTION

In Section 3.2, we discussed the basic concept of an efficient test pattern generator design using a compatibility matrix for a given test set. With the given theory, we understood that by analyzing the relationship of inputs in a test set, we can reduce the size of a pseudo-exhaustive test pattern generator to a more compact size while maintaining the same fault coverage. The compatibility matrix concept is an effective technique especially when there are a lot of logic 'X' (i.e., don't cares) in the test vectors. In this case, its input reduction rate increases dramatically because logic 'X' can be treated as either logic '0' or logic '1' when defining compatibility relationships. A detail algorithm for input reduction using a compatibility matrix concept is shown below.

1. Fetch a complete test set and store those data in a 2-D array.

2a. Analyze reference column X and compare with column Y to see if they are compatible. Increment counter by 1 if their test bits are equal and loop through until it reaches the last test vector. If any incompatible bits are found along the way, break out of the loop and reset the counter back to 0.

2b. Analyze reference column X and comparing column Y for inversely compatible relation. Increment counter by 1 if their test bits are opposite to each other and loop through until it reaches the last test vectors. In any compatible bits are found along the way, break out of the loop and reset the counter back to 0.

3. If we find either compatible or inversely compatible inputs from step 2, combine those inputs to one test signal. If there is any logic 'X' values in a reference column, recalculate and replace the logic 'X' value with the corresponding one that is explained in Table 1 and 2. Lastly, replace comparing column with '3' to indicate that its column has already been merged with others.

Table 1: If two primary inputs are compatible relation

Initial Value		Final Value	
Column X (Reference Column)	Column Y (Comparing Column)	Column X (Reference Column)	Column Y (Comparing Column)
X	0	0	3
X	1	1	3
X	X	X	3

Table 2: If two Primary Inputs are inversely compatible relation

Initial values		Final values	
Column X (Reference Column)	Column Y (Comparing Column)	Column X (Reference Column)	Column Y (Comparing Column)
X	0	1	3
X	1	0	3
X	X	X	3

4. Repeat step 2 and step 3 for column Y (Comparing Column) and loop through until the last primary input.

5. Find the next column X (Reference Column) that has not been merged yet with others and repeat step 2 to step 4.

6. Repeat step 2 to step 5 until column X (Reference Column) reaches the total number of columns in a test set.

3.4 EXPERIMENTAL RESULTS

We performed experiments using the compatibility matrix algorithm for two different test scenarios. Table 3 shows the results for the scenario where the test set for each benchmark contains an average number of logic ‘X’ values in the test vectors, and

Table 4 shows the results for the scenario where the test set for each benchmark contains a lot of logic 'X' values in the test vectors. Note that the number of test vectors goes up in Table 4 when the number of X's is increased. As can be seen in the results, the input reduction rate is heavily depended on the number of logic 'X' as we have discussed earlier in Section 3.3. By comparing and analyzing the two different test scenarios, we are able to perceive a weakness for the compatibility matrix algorithm and understand more clearly what other conditions could be added to compensate for this issue.

As we have discussed in Section 3.3, a test set that contains a lot of logic 'X' values in the test vectors results in higher input reduction compared to the test set when it contains a smaller number of logic 'X' values in the test vectors. This difference is indeed seen in the results in Tables 3 and 4. If no input reduction is used, then there is one test bit for each primary input in the circuit, thus the number of initial test bits is shown in the fourth column in Tables 3 and 4. And the final number of required after applying input reduction using the compatibility matrix approach is shown in the fifth column. These are compared in the sixth column. As can be seen from the Table 3 and 4 results, the size of the test pattern generator can be reduced by 34% to 93% using this method.

Table 3: Using compatibility matrix for test sets containing average number of logic 'X'

Circuit	Number of Primary Inputs	Number of Test Vectors	Initial Width	Final Width	Width Reduction (%)
s5378.x	214	111	214	136	36.45
s9234.x	247	159	247	162	34.41
s13207.x	700	236	700	212	69.71
s15850.x	611	126	611	275	54.99
s38417.x	1664	99	1664	538	67.67
s38584.x	1464	136	1464	702	52.05

Table 4: Using compatibility matrix for test sets containing a lot of logic ‘X’

Circuit	Number of Primary Inputs	Number of Test Vectors	Initial Width	Final Width	Width Reduction (%)
s5378.32s	214	196	214	44	79.44
s9234.64s	247	205	247	97	60.73
s13207.44s	700	266	700	52	92.57
s15850.46s	611	269	611	64	89.53
s38417.100s	1664	376	1664	113	93.21
s38584.100s	1464	296	1464	146	90.03

3.5 DRAWBACK

The major drawback of using the conventional compatibility matrix algorithm is that this algorithm is heavily dependent on the number of logic ‘X’ in the test vectors. If a test set only contains logic ‘0’ or logic ‘1’, it is unlikely to identify many compatible and inversely compatible relations among the primary inputs. As mentioned previously, the compatible matrix approach is more effective as the test vectors contain a greater number of logic ‘X’ because logic ‘X’ value gives us the extra option to choose either logic ‘0’ or logic ‘1’ when evaluating compatibility relationships. The probability of merging primary inputs goes up with more logic ‘X’ values in the test set. In the next chapter, we are going to propose a different way of performing input reduction that is not as dependent on the number of logic ‘X’ values and thus can compensate for the weakness part of compatibility matrix algorithm.

Chapter 4: Input Reduction Technique Based on Linear Operations

4.1 Idea of Using Linear Operations in Pseudo-Exhaustive Testing

As we mentioned in Chapter 3, the conventional on-chip test pattern generator design technique using input reduction based on a compatibility matrix has one critical drawback which is that its performance is heavily depended on the number of logic 'X' values in the test vectors. A new technique based on linear operations is proposed here which can solve this problem. By using linear operations, we are able to perform input reduction while handling the major drawback of the previous algorithm.

The key idea in the proposed method is to systematically identify inputs which can be driven by linear combinations of the other inputs. This can be done by treating the test set as a set of Boolean vectors that span a linear subspace, and then finding a basis for that linear subspace. Test signals are only needed for each primary input in the basis while all other primary inputs can be generated as linear combinations of the test signals. This corresponds to a pseudo-exhaustive test pattern generator whose width is equal to the number of primary inputs in the basis, and then adding XOR gates to combine these test signals together to drive the remaining primary inputs.

One complication that arises is handling 'X' logic values in the test set. Each value in the test set is required to be either logic '0' or logic '1' to form a basis, and this is a problem for our case since we should allow test vectors to contain logic '0', '1', and 'X'. There is a way that we can address this issue by following a certain rule which we will discuss further in Section 4.3.

4.2 Using Gauss-Jordan Elimination to Find Basis

To identifying linear combinations of inputs that can be used to drive another input, the test set matrix can be considered as a linear subspace. For example, in Figure 5, the transpose of the test matrix is shown where each row corresponds to a primary input, and each column corresponds to a test vector. If we treat this as a linear subspace, then Gauss-Jordan Elimination can be used to find a basis for the linear subspace. Gauss-

Jordan Elimination creates one pivot in each column. Every vector in the subspace can be generated by taking a linear combination of the basis vectors. In Figure 5, the matrix on the right shows a set of 4 pivot vectors which were obtained by performing Gauss-Jordan Elimination. As can be see, the pivot vectors depend only on I_1, I_2, I_3, and I_6. This means that I_4 and I_5 can be expressed as linear combinations of a subset of those 4 inputs. In particular, I_4 is the same as the first pivot vector which is equal to I_1. I_5 is equal to the XOR of the first and third pivot vector which is equal to ($I_1 \oplus I_3$). Using a compatibility matrix would only have identified I_1 and I_4 as being directly compatible, so would achieve an input reduction of 6 down to 5. Whereas the proposed method would achieve an input reduction of 6 down to 4 since it would need only 4 test signals for I_1, I_2, I_3, and I_6, and would generate I_4 and I_5 from those. The pseudo-exhaustive test pattern generators for using the compatibility matrix and using the proposed method are shown in Fig. 6-7.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{matrix} I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \\ I_6 \end{matrix} \rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} I_1 \\ I_1 \oplus I_2 \oplus I_3 \\ I_3 \\ I_3 \oplus I_6 \end{matrix}$$

Figure 5: Forming Basis using Gauss-Jordan Elimination

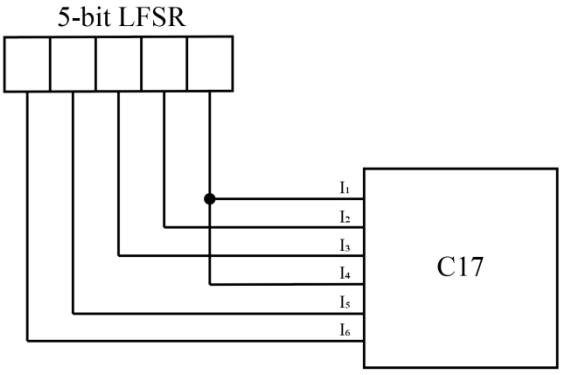

Figure 6: Pseudo-exhaustive Test Pattern Generator design using compatibility matrix

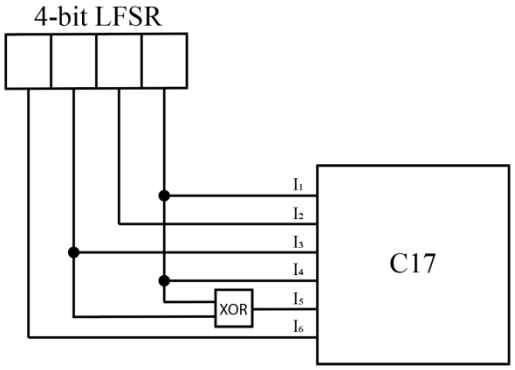

Figure 7: Pseudo-exhaustive Test Pattern Generator design using linear operations

4.3 Algorithm For Input Reduction Using Linear Operations

Now, we are going to present an algorithm for implementing input reduction using linear operations. The detailed algorithm for constructing the on-chip test pattern generator is shown below.

1. Fetch a test set and store those data in a 2-D array.
2. Transpose the test set.

The column of given each test set represents information of primary inputs and row of given test set represents information of test vectors. Because our goal is to reduce the total number of primary inputs in each test set using Gauss-Jordan Elimination method, we need to transpose the test set.

3. Find the first logic ‘1’ in a column and copy the entire row into new 2-D array. Perform XOR operation for the entire row with the pivot row if other rows in the column also have logic ‘1’. If the other rows have logic ‘X’, then simply set to logic ‘0’.

4. If there is no logic ‘1’ that is being used as pivot for a column, set the first logic ‘X’ that we find to logic ‘1’ so that it can be used as a pivot for the corresponding column. Store the entire row information into next position in the new 2-D array and perform XOR operation with each row that was already stored in the new 2-D array to convert as an identity matrix. Set rest of logic ‘X’ to logic ‘0’ if there is more logic ‘X’ in the other rows.

5. Perform step 3 and step 4 continuously until we reach the last primary input column.

Applying linear transformation theory can provide powerful input reduction for fully specified data. However, it cannot handle logic ‘X’ values since linear transformation does not allow us to have any unknown values. Given that we can use logic ‘X’ as either logic ‘0’ or logic ‘1’, we can appropriate set logic ‘X’ values to specified values as explained in the above algorithm.

4.4 Experimental Results

We tested the proposed approach for input reduction with linear operations on two different scenarios similar to what was done in Section 3.4. Table 3 shows the results for the scenario where the test set for each benchmark contains an average number of logic ‘X’ values in the test vectors, and Table 4 shows the results for the scenario where the test set for each benchmark contains a lot of logic ‘X’ values in the test vectors.

As we mentioned previously, the proposed approach can somewhat compensate for the drawback of using the compatibility matrix method and its effect is clearly shown in Table 5 and Table 6. We have seen that using a compatibility matrix was not showing great performance for input reduction when a test set contains a small number of logic ‘X’ values. However, the proposed approach is effective on these test sets.

Table 5: Using Gauss-Jordan Elimination for test sets containing average number of logic 'X'

Circuit	Number of Primary Inputs	Number of Test Vectors	Initial Width	Final Width	Width Reduction (%)
s5378.x	214	111	214	111	48.13
s9234.x	247	159	247	159	35.63
s13207.x	700	236	700	236	66.29
s15850.x	611	126	611	126	79.38
s38417.x	1664	99	1664	99	94.05
s38584.x	1464	136	1464	136	90.71

Table 6: Using Gauss-Jordan Elimination for test sets containing a lot of logic 'X'

Circuit	Number of Primary Inputs	Number of Test Vectors	Initial Width	Final Width	Width Reduction (%)
s5378.32s	214	196	214	173	19.16
s9234.64s	247	205	247	205	17.00
s13207.44s	700	266	700	257	63.29
s15850.46s	611	269	611	269	55.97
s38417.100s	1664	376	1664	376	77.40
s38584.100s	1464	296	1464	296	79.78

4.5 DRAWBACK

The proposed method of input reduction with linear operations is relatively more effective when the test set contains fewer logic 'X' values. However, it can be less effective when a test set has a lot of logic 'X' values. To address this, it can be combined with the conventional approach of using a compatibility matrix as will be discussed in the next chapter.

Chapter 5: Combined Method

5.1 Introduction for Combined Method

In Chapter 3, we discussed that we can achieve input reduction for a test set using the conventional compatibility matrix algorithm. It provides better input reduction for test sets containing lots of logic 'X' values, but is less effective for test sets containing few logic 'X' values. In Chapter 4, we proposed a methodology for performing input reduction using linear combinations. The method can also be used to compress a test set. The proposed method is most effective when the test set has a small number of logic 'X' values. So we ultimately decided to combine both algorithms together to get the best overall input reduction regardless of the characteristics of the test set.

5.2 Combined Algorithm

We can first use the compatibility matrix approach (as described in Section 3.3) to achieve as much input reduction as possible, and then use the proposed algorithm based on linear operations (as described in Section 4.3) to try to achieve further input reduction.

5.3 Experimental Results

Table 7 and Table 8 show the results for the combined method. Clearly, it achieves better or equal input reduction than using only one of two methods individually. Table 7 shows that the combined method is better in all cases. In Table 8, in comes cases, the final result ended up the same as using the compatibility matrix by itself because for those testbenches there were so many logic 'X' values that the compatibility matrix method was able to maximally optimize the input reduction.

Table 7: Using combined method for test sets containing average number of logic ‘X’

Circuit	Number of Primary Inputs	Number of Test Vectors	Initial Width	Final Width	Width Reduction (%)
s5378.x	214	111	214	111	48.13
s9234.x	247	159	247	159	35.63
s13207.x	700	236	700	87	87.57
s15850.x	611	126	611	93	84.78
s38417.x	1664	99	1664	99	94.05
s38584.x	1464	136	1464	136	90.71

Table 8: Using combined method for test sets containing a lot of logic ‘X’

Circuit	Number of Primary Inputs	Number of Test Vectors	Initial Width	Final Width	Width Reduction (%)
s5378.32s	214	196	214	44	79.44
s9234.64s	247	205	247	97	60.73
s13207.44s	700	266	700	52	92.57
s15850.46s	611	269	611	64	89.53
s38417.100s	1664	376	1664	113	93.21
s38584.100s	1464	296	1464	146	90.03

We have included Figure 6 and Figure 7 to provide more direct comparison results for each method. The Y-axis represents the percentage of input reduction that was achieved from each initial test set, while the X-axis has one entry for each different test set. Lower number of percentage in Y-axis indicates that it has less compression rate and higher number of percentage indicates that it has higher compression rate for each test set.

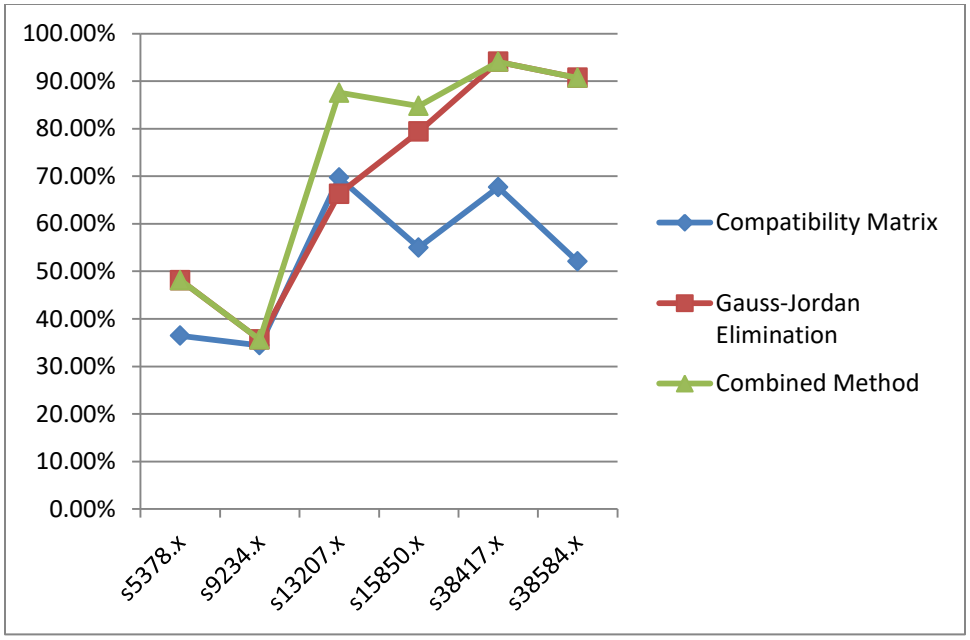

Figure 6: Graphical Representation of Compression Rate with average number of logic 'X'

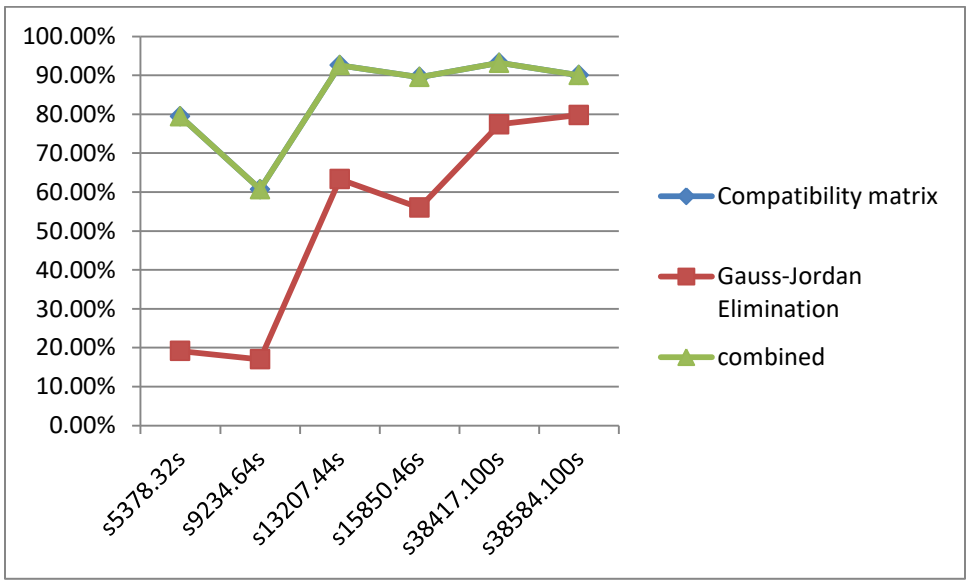

Figure 7: Graphical Representation of Compression Rate with a lot of logic 'X'

Chapter 6: Conclusion

Applying an exhaustive test set for testing a circuit is generally not a feasible approach due to having exponential test time with respect to the number of primary inputs. However, pseudo-exhaustive testing can achieve the same goal of detecting all combinational faults with a much shorter test. Input reduction has been proposed in the past as a way to achieve a practical test length for pseudo-exhaustive testing. This thesis proposed a new methodology for input reduction based on using linear combinations. It was shown that this approach is able to further increase the amount of input reduction that can be achieved compared with the conventional approach of using a compatibility matrix. Furthermore, it was shown that the proposed approach is effective even when the numbers of 'X' values in the test set is small. It was also shown that the proposed method can be used together with a compatibility matrix to achieve good results regardless of the characteristics of the test set. Pseudo-exhaustive testing with input reduction offers a number of advantages in terms of thorough testing, no need for an external tester, and avoiding the test time bottleneck of needing to bring data through the chip pins.

Bibliography

[1] M. Abramovici, M. A. Breuer, and A. D. Friedman, "Digital Systems Testing and Testable Design," Piscataway, New Jersey: IEEE Press, Revised Printing, 1994.

[2] M. L. Bushnell and V. D. Agrawal, "Essentials of Electronic Testing for Digital, Memory & Mixed-Signal VLSI Circuits," New York: Springer Science, 2000.

[3] L. -T. Wang, C. -W. Wu, and X Wen, Eds., "VLSI Test Principles and Architectures: Design for Testability', San Francisco: Morgan Kaufmann, 2006.

[4] X. Wen, "VLSI Testing and Test Power," *2011 International Green Computing Conference and Workshops*, Orlando, FL, 2011, pp. 1-6.

[5] T. Kirkland and M. R. Mercer, "Algorithms for automatic test-pattern generation," in *IEEE Design &Test of Computers*, vol. 5, no.3, pp. 43-55, June 1988.

[6] E. J. McCluskey, "Verfication Testing – A Pseudoexhaustive Test Technique," *IEEE Transactions on Computers*, C-33(6): 541-546, June 1984.

[7] S. S. Yau and Yu-Shan Tang, "An Efficient Algorithm for Generating Complete Test Sets for Combinational Logic Circuits," in *IEEE Transactions on Computers*, vol. C-20, no. 11, pp. 1245-1251, Nov. 1971.

[8] Chih-Ang Chen and S. K. Gupta, "A methodology to design efficient BIST test pattern generators," *Proceedings of 1995 IEEE International Test Conference* (ITC), Washington, DC, 1995, pp. 814-823.

[9] Chih-Ang Chen and S. K. Gupta, "Efficient BIST TPG design and test set compaction via input reduction," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, no. 8, pp. 692-705, Aug 1998.

Vita

Kangjoo was born in South Korea. His technical study area is integrated circuits and systems and earned his Bachelor's degree in Electrical Engineering at the University Texas at Austin at May 2016. He is currently pursing Master of Science in Electrical Engineering with Thesis program at the University of Texas at Austin.

Permanent address: 2808 Wisdom Creek Dr. Flower Mound, Texas, 75022

This thesis was typed by Kangjoo Lee.