

STUCK-AT-FAULT TEST SET COMPACTION

A Senior Honors Thesis

by

JASON MICHAEL VANFICKELL

Submitted to the Office of Honors Programs
& Academic Scholarships
Texas A&M University
in partial fulfillment of the requirements of the

UNIVERSITY UNDERGRADUATE
RESEARCH FELLOWS

April 2004

Major: Computer Engineering

STUCK-AT-FAULT TEST SET COMPACTION

A Senior Honors Thesis

by

JASON MICHAEL VANFICKELL

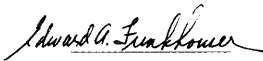
Submitted to the Office of Honors Programs
& Academic Scholarships
Texas A&M University
in fulfillment of the designation of

UNIVERSITY UNDERGRADUATE
RESEARCH FELLOW

Approved as to style and content by:



M. Ray Mercer
(Fellows Advisor)



Edward A. Funkhouser
(Executive Director)

April 2004

Major: Computer Engineering

ABSTRACT

Stuck-at-fault test set compaction

(April 2004)

Jason Michael VanFickell
Department of Electrical Engineering
Texas A&M University

Fellows Advisor: Dr. M. Ray Mercer
Department of Electrical Engineering

Proper testing of manufactured digital circuits is critical to ensuring the number of defective parts is minimized. Automated test pattern generation tools are created in order to produce test patterns that can be applied with the intention of identifying as many defective parts as possible. The increasing complexity of digital circuit designs causes this task to continue to increase in difficulty. At the same time, the amount of time dedicated to testing should be kept constant. Therefore, it is crucial to limit the number of test patterns that are applied to any given circuit. Additionally, tester memories may limit the number of test patterns that may be applied at one time. This research demonstrates several existing methods of compaction and introduces a new method for measuring the contribution of each test pattern. Both static and dynamic compaction methods were implemented and evaluated in terms of final test pattern set size and diversity of excitation. The program resulting from this research has been shown to equal or surpass an existing automated test pattern generation tool.

ACKNOWLEDGMENTS

I would like to thank my honors fellows advisor Dr. M. Ray Mercer for his continuous support and advice throughout my undergraduate research experience.

I would also like to thank Jennifer Dworak, Jimmy Wingfield, Brad Cobb, Dr. and Sooryong Lee for their contributions, ideas, and feedback. I would especially like to thank Mike Trinka for his contribution of the original SuperDA fault simulator and Justin Ray for his work on the implication component of the SuperDA test pattern generator.

I would also like to thank the Department of Electrical Engineering at Texas A&M University for providing me with the opportunities and financial support that helped me to complete this project.

Finally, I would like to thank God, my creator and savior, for all of the blessings He has given me and for providing me with inspiration, opportunities, and strength.

TABLE OF CONTENTS

	Page
ABSTRACT	iii
ACKNOWLEDGMENTS	iv
TABLE OF CONTENTS	v
LIST OF FIGURES	vi
LIST OF TABLES	vii
INTRODUCTION	1
Fault models	2
Fault Detection	3
Excitation Balance	3
Pattern Metric	4
METHODOLOGY	7
Program Flow	8
Fault Target Selection	9
Redundancy Elimination	9
Redundancy Creation	10
Dynamic compaction technique	11
Assignment Justification	11
Pattern Metric	12
RESULTS AND ANALYSIS	14
Dynamic Compaction Techniques	14
Fault Target Selection	14
Pattern Metric for Static Compaction	15
Pattern Metric for Dynamic Compaction	16
Comparison with another ATPG tool	18
CONCLUSION	20
REFERENCES	21
VITA	22

LIST OF FIGURES

	Page
Figure 1. Formula for computation of the excitation balance	4
Figure 2. Formula for computation of the pattern metric	5
Figure 3. High-level SuperDA program flowchart.....	8
Figure 4. Pattern Score and Fault Detections before compaction	13
Figure 5. Pattern Score and Detections after compaction.....	13
Figure 6. Test set size under pattern score based dynamic compaction	17
Figure 7. Pattern Score distribution with pattern metric dynamic compaction.....	18

LIST OF TABLES

	Page
Table 1. Fault Target Selection Methods.....	15
Table 2. Static Compaction using Pattern Metric.....	16
Table 3. Performance comparison of Gulf and SuperDA.....	19

INTRODUCTION

As digital integrated circuits continue to increase in complexity, the task of testing the manufactured chips resulting from fabrication has become an increasingly difficult problem. The expectation that Moore's law will continue to be upheld means that this problem will not subside any time soon [1].

The first reason for this increased difficulty is the greatly increased number of gates in the designs. As the complexity of the circuit increases, the number of points of failure increases also. The yield rate can be expected to decrease under the influence of this external trend. If the yield decreases due to a greater number of total faults, the task of identifying the faulty manufactured chips becomes far more crucial than before.

Another reason integrated circuit testing will continue to become more difficult is the gap between the number of fault points inside a chip and the number of input (excitation) and output (observation) opportunities. The number of outside connections on chips has not grown nearly as fast as the number of transistors. The continuation of this trend will result in a shortage of communication between the chip and circuit testing equipment. While this communications gap does not inhibit the design process or the functionality of digital circuits, it increases the difficulty of preserving favorable defective part levels.

By generating test patterns in a way which maximizes the value of each individual test pattern, two valuable objectives may be achieved. First, depending on the accuracy of the fault model used, the coverage of the true manufacturing defects may be increased. Second, the number of test patterns necessary to achieve a desired defective part level may be decreased.

This thesis follows the style and format of *IEEE Transactions on Automatic Control*.

Achieving the two objectives of low test pattern size and high defect coverage is the key to performing exceptional test pattern generation. Defect coverage must be high in order to minimize the number of defective chips shipped. Test pattern sets should be kept to a limited size in order to keep the amount of time that must be devoted to testing economically reasonable.

The fundamental reason that automated test pattern generation is such a challenge is that it fundamentally is an NP-complete problem. Given an exponentially high amount of time, the task of determining all possible tests for a circuit could be completed. However, in the practical environment of industry, the task of testing must be done much faster. Even if complete information about all test patterns was readily available, a covering problem remains if the number of test patterns is reduced to a more reasonable value facilitating testing in a practical amount of time.

This research study was intended to examine various methods for altering an ATPG (Automated Test Pattern Generation) tool with the goal of improving test set compaction. A new metric for evaluating the value of a test pattern is evaluated in terms of its ability to facilitate further compaction of test pattern sets.

Fault models

Because of the esoteric nature of the actual defects which occur during manufacturing and their heavy dependency upon the type of technology used, it is highly preferable to use abstract fault models instead. Once a fault model is chosen, the faults within that model are assumed the only points of failure that the test patterns should need to be able to detect. Beside the advantage of technology abstraction, alternative fault models also greatly reduce the number of faults under consideration, which helps to improve the computational complexity of the ATPG process.

The stuck-at fault model is the primary model used for static testing. Implementation of this model is simple, since it only requires a particular point in the circuit to be forced to a logic one or a logic zero. Previously, the level of stuck-at-fault coverage achieved by a test pattern set was considered a good indicator of the

effectiveness of the test pattern set as a whole. This model has been superseded more recently with the concept of multiple stuck-at-fault detections. The defective part level has been shown to be further reduced when each fault is required to be detected multiple times. It was shown that the higher the required minimum number of detections, the lower the defect part level becomes [2].

An optimal number of minimum detections was determined to be 15 by prior research with real manufactured digital circuits [3]. While this number of detections of each fault may not be perfectly optimal for all circuits or all fault models, it will be used throughout the scope of this research. This provides a sufficient degree of multiple fault detection in order to evaluate the proposed methods in the scope of multiple detection fault targeting. At the same time, the number is kept sufficiently low to allow the ATPG process to be completed in a reasonable amount of time.

Fault Detection

In order for a fault to be considered detected under the stuck-at-fault model, two conditions must be satisfied: excitation and observation. The fault site is excited whenever applying a known set of input assignments to the circuit will guarantee that the value at the site of the fault will be the opposite of the value of the stuck-at-fault. This condition is necessary, but not sufficient for ensuring the detection of a fault.

Observation, on the other hand, is the propagation of the faulty state to primary outputs. It is only necessary for a change in the value of the fault site to be detectable on one or more of the primary outputs. Therefore, the imposed observation conditions for a pair of complimentary stuck-at-faults at the same site should in fact be the same. However, the conditions for excitation for the same pair of faults may be different and thus require the method of observation to vary.

Excitation Balance

The abstraction of the circuit defects to the fault model is a leaky abstraction at best. Therefore, one attempt at bridging the gap between the model and reality has built

upon the concept of multiple stuck-at-fault detections. The effectiveness of an ATPG tool may be measured is the degree of excitation balance existing in the test pattern set. Excitation balance may be computed using the formula shown in Figure 1, devised by Dworak [4]:

$$Exc_Bal_j = 0.25 - \frac{\sum_{i=1}^{\# \text{ of sites}} (0.5 - \text{ones_prob}_{ij \text{ detected}})^2}{\# \text{ of sites}}$$

Figure 1. Formula for computation of the excitation balance

Excitation balance is a metric computed for every detectable fault in the entire circuit. It is intended to provide a measure of the degree to which the patterns detecting the same fault differ. This measure insures against misuse of the underlying fault model. One example of abuse would exploit a pattern with four unassigned inputs remaining. A cunning ATPG tool could create 16 nearly identical patterns simply by expanding upon the 2^4 possible combinations for the unassigned inputs. This would allow a single detection to be instantly transformed into multiple detections. However, the resulting patterns would have very little differentiating each other. The intention of finding a multiple stuck-at-fault detection test set is to increase the degree of excitation the circuit undergoes without moving to a more complex fault model. Allowing patterns which are nearly identical adds very little value and can be expected to have less of an impact upon the defective part level than truly unique detecting patterns. In order to simplify the analysis, only the median of the excitation balance values for all faults is considered. The formula is constructed so that higher values indicate a greater degree of balance in the excitation of faults.

Pattern Metric

An external measure of test set compaction is simply the size of the final test set, which includes all patterns necessary to suitable detect all faults in the fault model

chosen. While trivial to determine, this simplistic measure provides little value for the purpose of attempting to improve the test pattern generation process in order to arrive at a particular level of compaction. During the test generation process no meaningful data can be derived from the number of test patterns since some faults have had their requirements met, and others have not. Furthermore, some faults are much more difficult to observe than others. A truly comprehensive measure would be able to take these transient factors into account in order to provide the ability to measure the value of a test set during the process of test pattern generation.

The metric shown in Figure 2 is proposed to provide a score indicating the relative value of any given test pattern. In the given formula, which is computed for a given test pattern identified by the variable i , the element detections_j refers to the number of detections made by fault i , which is one of the faults detected by pattern j .

$$\sum_{j=0}^{\text{detections}_j} e^{-\text{detections}_j}$$

Figure 2. Formula for computation of the pattern metric

It can be implied from this formula that as the number of test patterns applied continually increases past the number necessary for detecting all faults, the typical value of any additional test pattern according to the given formula will approach zero. Furthermore, in a test pattern set in which traditional static test compaction has been applied to remove all redundant patterns, the minimum possible value of a test pattern must be the inverse of the minimum desired number of detections for each fault. Therefore, it is evident that higher scores are better, and the nominal value for a test pattern approaches zero.

Gaining insight from the distribution of pattern scores may provide opportunities to improve test set compaction. Since patterns with higher scores are assumed more

valuable, the compactness of the test set might be improved when the scores of the patterns in the test set improve as a whole.

There were two guiding principles observed in the implementation of the algorithm for compact TPG. First, the toughest faults to find tests for were always targeted first. Second, each pattern is sought to be made as useful as possible.

Because the majority of faults in the stuck-at-fault model are easy to find, the vast majority of easy faults are detected multiple times because of fortuitous detection. For the tough faults, however, imposing the requirement of multiple detections can increase the effectiveness of the generated test patterns. As a result, the desired number of detections may be increased, or the size of the test set can be reduced. These are two contradicting objectives; the focus of this research is the latter objective.

METHODOLOGY

The starting point for the implementation of these ideas was an existing logic fault simulator previously developed at Texas A&M University by Mike Trinka. This fault simulator, which was named SuperDA (Super Defect Analyzer), was originally used to perform research on fault analysis using fault and defect signatures. One major component of this project was a fault simulator which worked on the gate level. A fault simulator is able to take a circuit definition and a set of test patterns as input and produce a list of all detected faults according to the defined fault model. Because this implementation already contained an optimized fault simulator, it was chosen as the basis for the automated test pattern generator. One of the advantages of this approach is that a fault simulator was already available to verify the correctness of the pattern generator under development.

One novel idea behind this new test pattern generator was the use of logic implication to deduce the input assignments necessary in order to both excite and observe a given fault. However, due to complications in the circuit such as reconvergent fanout [1], the set of assignments yielded from implication may not always lead to fault detection. For this reason, a set of procedures based on the Podem method [5] was implemented in order to recursively search a subset of the search space for a valid test detecting a particular fault. The ability to store all test patterns and the faults they detected was also incorporated into the SuperDA program. Thus, a complete SuperDA was created using a fault simulator as a basis.

In order to test new compaction techniques, various methods were implemented into SuperDA. Both static and dynamic compaction have been implemented so that both types of methods could be compared with each other. The hypothesis was that changes to the ATPG process itself could more effectively produce smaller test set sizes than static compaction alone without introducing as additional computational effort.

Program Flow

The program flow within SuperDA is primarily managed by a hierarchy of loops. The loops are constructed in such a way that the desired minimum level of fault detection may be met, or so that the value of each test pattern may be maximized. A high-level overview of the ATPG process employed by SuperDA is depicted in the flow chart of Figure 3.

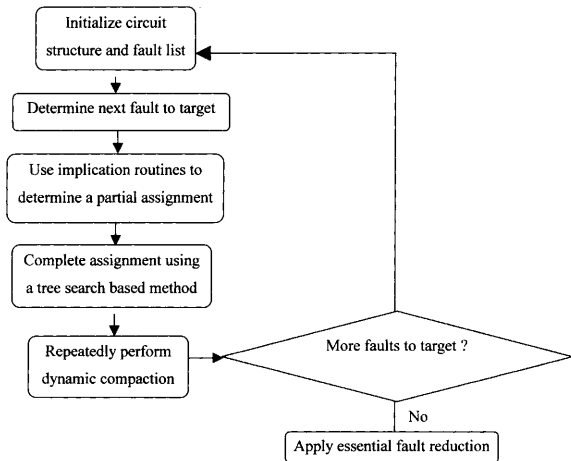


Figure 3. High-level SuperDA program flowchart

Fault Target Selection

One critical stage of any ATPG system is the selection of the fault or faults which are to be targeted. While it is a minor aspect of the ATPG system as a whole, fault target selection can greatly influence the speed and effectiveness of the tool as a whole. Because most easy faults are assumed to be detected fortuitously, little benefit would be gained from targeting these types of faults directly. Much greater overall test set compaction could be achieved by allowing these faults to be fortuitously detected while other more difficult faults are actually being targeted. One common method for determining which fault should be targeted is to find the least detected fault. During the ATPG process, one measure of the difficulty of a fault is the number of times it has already been detected. Early in the ATPG process, this measure of fault difficulty will not be accurate. However, it is expected to progressively improve during the ATPG process.

Other potentially beneficial methods may exist for determining the next target fault. An evaluation of each of these would allow the particular method which is more efficient in terms of both time and the degree of test set compaction to be chosen. Two methods have been implemented and evaluated within the SuperDA project. The first is a complete random ordering, which establishes a neutral basis since it has no deterministic notions. The second sort the faults according to their detection counts.

Redundancy Elimination

The most basic form of static compaction relies on the identification and removal of test patterns which are not required for the goal number of detections for each fault to be achieved. This research has been focused on the creation of multiple-detection test sets where the desired number of detections is fifteen. Therefore, any pattern for which all detected fault have fifteen or more detections may be declared redundant. Redundant patterns may be safely removed from the set of test patterns.

Redundancy Creation

A more complex method of compaction builds upon the concept of an essential fault. An essential fault as defined by [6] is a fault detected only once. The cited research evaluated a new method named Essential Fault Reduction. This method constructed and utilized a list of compatible faults in order to prune essential faults from existing test patterns, thereby creating a redundant test pattern.

The ATPG algorithms used for this research have been evaluated with respect to the ability to detect all possible faults multiple times. The concept of an essential fault translates well into this case. Modifying another pattern in the test set to increase the number of detections of an essential fault to two will still cause the fault to cause being essential. Rather than dealing with the complexity of computing an independent fault set as performed by the methods in [6], the implemented algorithm instead takes existing patterns and uses the same tree-based search method while targeting other potentially detectable essential faults for detection. If any of the targeted faults become successfully detected, then the extended test pattern replaces the original test pattern in the test set, and a redundancy is known to have been produced.

All patterns which become redundant may be removed one at a time. However, the removal of one pattern may cause some of the faults it detects to become essential. Therefore, the order in which redundant patterns are removed could influence the number of patterns removable in the long run.

Since the process of redundancy creation greatly increases the number of redundancies, it is useful to investigate various possible approaches to redundancy removal. The order in which patterns are considered for redundancy can have an impact upon the effectiveness of static compaction; three ordering methods were evaluated. The first was the most simple: a random ordering of all patterns which is intended to provide a neutral basis for comparison. The second two methods are attempts to improve compaction by taking the value of each pattern into consideration. Patterns are

considered for removal earlier based upon either their detection counts or the pattern score metric discussed earlier.

Dynamic compaction technique

Dynamic compaction allows the number of patterns generated in the first place to be reduced, thereby improving compaction. Since an intuitive method for achieving compaction is to increase the number of detections made by each individual pattern, a pattern-based method was employed.

The following type of method has been used before to improve test set compaction of stuck-at-fault test sets [6]. Pattern based dynamic compaction was first implemented by creating a loop which was entered immediately after a detection of a targeted fault had been made. Because in the majority of cases the detecting pattern had some inputs which had remained unassigned, this remaining flexibility in the pattern was seen as a venue for increasing the number of fault detections without increasing the cost in terms of the number of test patterns.

Assignment Justification

The underlying technique of dynamic compaction was supplemented with another routine which performed assignment justification. The first is related to one particular shortcoming in the method for completing inputs assignments in order to achieve a detection of a particular fault. The tree-based search method employed for finding detecting patterns exploits the parallel computation ability of the microprocessor by performing 32 simultaneous simulations of the circuit using bitwise operations at each step. While this provides an increase in speed of approximately thirty-two-fold, this can also result in unnecessary input assignments being made. If the freedom of each test pattern is desired to be maximized, then these superfluous assignments should be eliminated.

Another reason for justifying all input assignments is related to fault difficulty. Although in general the detection of more faults by a pattern is a good thing, it is truly

the detection of tough faults which is of value. Therefore, input assignments which ensure the detection of tougher faults should be kept, while input assignments which only provide for the detection of easier faults are unnecessary and should be removed in order to provide greater freedom for dynamic compaction.

The routine which is employed by SuperDA re-simulates test patterns, once for each primary input in the circuit. Each time, a different primary input assignment is temporarily disabled. If the detection of a difficult fault is lost as a result, then the assignment is reversed back to its previous value. This process is repeated for all assignments made in the pattern. In order to minimize the amount of deterministic bias, the ordering of inputs to iterate through is randomized prior to the start of the input justification process.

Pattern Metric

Two accepted practices in test set compaction came under question over the course of this research and led to new proposed methods for measuring characteristics of both test patterns and faults. The first such idea was the concept of a metric, which could be easily calculated and was capable of giving a rough indication of the value of a single test pattern. This would accomplish two objectives. First, it would provide a measure for other compaction techniques to use in order to rank patterns in terms of their contribution to fault coverage. Second, it could allow for a quantitative measure of the compactness of a test pattern set beyond the aggregate number of test patterns. This is the simplest known measure of test pattern effectiveness relative to fault coverage other than the number of detections made by the pattern. Because faults vary significantly in difficulty, it is speculated that a more comprehensive measure could provide significant additional benefit.

A view of the correlation between the scores of patterns according to this metric and the number of detections made by a pattern is interesting to investigate. Figure 4 shows this correlation of the ATPG algorithm with all compaction disabled. Figure 5

shows the same correlation after static compaction techniques were applied to the same set of test patterns in order to significantly reduce the number of test patterns.

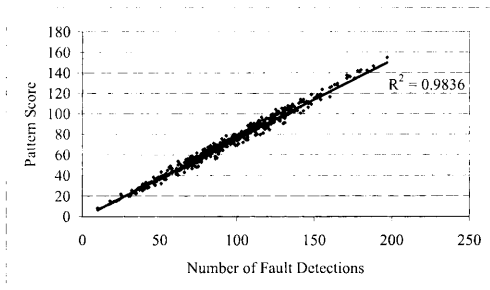


Figure 4. Pattern Score and Fault Detections before compaction

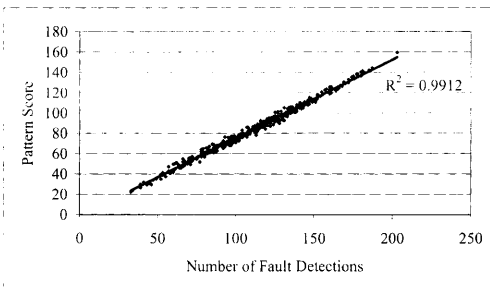


Figure 5. Pattern Score and Fault Detections after compaction

RESULTS AND ANALYSIS

SuperDA was used to form complete 15-detect test sets for several circuits. These circuits were chosen from a set of benchmark circuits established the 1985 IEEE Symposium on Circuits and Systems [7], and the same set of benchmark circuits have been used for in a variety of other research papers. It is useful to use this set of standard benchmarks as it allows research results to be easily compared with related research findings.

Each fault was targeted for detection exactly 15 times. For all of the circuits for which results were posted, every detectable stuck-at-fault is indeed detectable 15 times. This does not necessarily always have to be true, but it is an assumption that simplifies the algorithm which must be used to find a complete set of test patterns.

There are some stuck-at-faults in the benchmark circuits which are not detectable by any test patterns at all. This is most often due to a redundancy in the gate logic which through additional logic minimization could have been removed. In practice however, some redundancies are intentionally placed within a logic circuit in order to satisfy certain timing requirements.

Dynamic Compaction Techniques

An evaluation of the impact of the various dynamic compaction techniques is useful in order to determine if they are effective enough to justify any increase in computational time necessary. First, several of the benchmark circuits were tested with SuperDA with all forms of dynamic compaction disabled. Then, the circuits were tested with the dynamic compaction methods mentioned earlier, both with and without input justification.

Fault Target Selection

The method by which the next fault to be targeted is selected can have a significant impact upon the amount of CPU time necessary for ATPG and the total

number of test patterns produced. Two different methods were implemented within SuperDA and several of the benchmark circuits tested in order to evaluate and compare their effectiveness and speed. The results in Table 1 clearly show that selecting the fault target with the minimum detection count usually results in a test set size equal to or less than the corresponding size resulting from a random fault target selection method. Additionally, the affect on excitation balance is insignificant.

Table 1. Fault Target Selection Methods

Circuit Name	Random		Minimum Detection Count	
	Excitation Balance	Pattern Count	Excitation Balance	Pattern Count
c432	0.136296	1152	0.137393	895
c499	0.191652	873	0.191554	808
c880	0.170293	2693	0.171926	2583
c1355	0.173310	1394	0.173381	1293
c1908	0.187486	1708	0.186800	1708
c2670	0.190484	5148	0.189875	999

Pattern Metric for Static Compaction

The technique used for redundancy removal was also slightly altered in order to investigate whether the order in which patterns are considered for removal by redundancy might have an effect on the amount of attainable compaction. An experiment was conducted using the redundancy creation and redundancy removal algorithms which took a set of uncompact vectors and performed compaction. Three methods were employed in this experiment. First, the patterns were considered in a randomized order. The order of consideration was also altered to an increasing order of detection count and an increasing score according to the pattern metric. It is shown in Table 2 that the method which considered patterns in a random order performed significantly better than the other two methods. It is suspected that this is another

scenario in which a randomized approach is able to locate opportunities most deterministic approaches are unable to find.

Table 2. Static Compaction using Pattern Metric

		Circuit Name			
		c432	c499	c880	
Starting number of patterns		1094	810	2638	
	Sorting method:				
Final Number of patterns	Random	CPU Time (s)	266.49	47.27	354.61
		Patterns	492	806	320
	Detection Count	CPU Time (s)	180.61	54.95	382.17
		Patterns	524	806	346
	Pattern Score	CPU Time (s)	184.58	25.28	406.67
		Patterns	525	806	343

Pattern Metric for Dynamic Compaction

The pattern score metric for quantitatively measuring the contribution of a single test pattern was integrated into SuperDA in order to determine if the metric could improve dynamic compaction. The algorithm, which ran with a given probability, located the pattern with the lowest score according to the metric and removed that pattern from the test set. This algorithm clearly could not be run with a 100% probability, or there would never be a net gain of test patterns and a complete test set would never result after an infinite amount of time. The resulting test set sizes as reported both before and after are shown by the graph in Figure 6 for probabilities ranging from 0% to 50%.

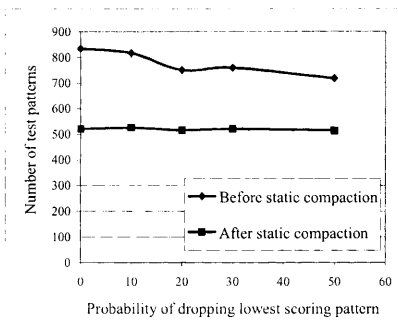


Figure 6. Test set size under pattern score based dynamic compaction

The resulting graph in Figure 6 clearly indicates that the number of test patterns was impacted as the probability of calling the routine increased, the number of test patterns remaining before static compaction was applied. After static compaction was applied, however, the number of test patterns remained relatively unchanged even as the probability parameter was varied.

It is interesting to note, however, that the algorithm was successful at its direct objective, which was to decrease the occurrence of lower scoring patterns. Figure 7 depicts the minimum, 3 quartiles, and maximum of the distribution of patterns scores after the completion of ATPG with various probabilities of invoking the compaction routine. It can be observed that the floor of the distribution increases as the probability parameter increases. This success, however, did not translate into a decrease in test set size.

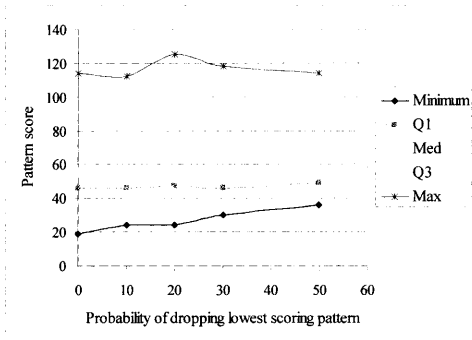


Figure 7. Pattern score distribution with pattern metric dynamic compaction

Comparison with another ATPG tool

Another similar ATPG tool called “Gulf” was developed by Dr. Sooryong Lee at the Computer Engineering Group at Texas A&M University prior to the implementation of test pattern generation in SuperDA. A comparison of these two tools is valuable in order to gain insight into the relative strengths and weaknesses of the methods behind each.

Gulf achieves test set compaction by using a greedy search algorithm for finding test patterns. This greedy algorithm was able to achieve significant compaction of 15 detect test sets by computing a weight for each possible branch in the tree based search for test patterns [8]. This weighting takes into account the importance of faults known to be detected and the probability of detecting additional faults with the set of input assignments already made.

SuperDA differs from Gulf in several respects. First of all, no greedy metric was applied; one fault was targeted at a time first using implication and then using a tree-based search method. Test set compaction was achieved using a pattern-based method which targeted additional faults sequentially. Table 3 shows a summary of the results from running both programs on various benchmark circuits.

For the circuit c432, SuperDA performed poorly and had a greater number of test patterns and a lower measure of excitation balance. For circuit c499, however, the measure of excitation balance in the patterns produced by SuperDA was significantly higher than those produced by Gulf with only a slight increase in the total number of test patterns. In circuits c1908 and c2670, both the number of necessary test patterns and the excitation balance were better in SuperDA.

It is believed that the relative performance of SuperDA improves with increased circuit size because the more random nature of the algorithm is more successful with larger search spaces than deterministic methods. The goal of Gulf is to maximize test set compaction by making wiser input assignments, and so the algorithm is more deterministic. This may impose more limitations on width of the search space typically explored, and in turn reduce the amount of diversity in test patterns. The process SuperDA uses is completely randomized, however, and so the search space would be expected to be more evenly explored.

Table 3. Performance comparison between Gulf and SuperDA

Circuit Name	GULF		SuperDA	
	Excitation Balance	Pattern Count	Excitation Balance	Pattern Count
c432	0.144461	496	0.128161	536
c499	0.135106	780	0.192135	789
c1908	0.185085	1661	0.186057	1658
c2670	0.156789	992	0.189676	940

CONCLUSION

The test pattern generation schemes integrated into the program known as SuperDA were capable of outperforming Gulf in two key areas. First, the measure of excitation balance which has been shown to be closely related to the defective part level was better in the majority of the circuits tested. Also, the number of test patterns was nearly the same or lower in all but one of the circuits. Two key differences exist between these two programs which may be attributed to this performance. SuperDA utilized an implication technique which was able to topologically analyze the circuit and determine preliminary sets of input assignments. Second, a much greater proportion of the decisions were made in a randomized fashion.

Although no direct applications for a metric measuring the value of each individual test pattern could be found, there are still other potential applications of the metric which could prove of value. The devised algorithms had their intended effect with respect to the pattern metric, but this success did not translate into improved test set compaction. An improved metric may be capable of resolving these difficulties. Perhaps the metric could be extended in such a way that the essential faults, or those which have the desired number of detections or less, could be given a much greater weight.

The type of dynamic compaction implemented in SuperDA was effective, but was unable to provide any benefit beyond the static compaction methods employed. Because the dynamic compaction methods also introduced additional computational complexity, static compaction based on the creation of redundant patterns is believed to be the more superior method for achieving maximal test set compaction.

REFERENCES

- [1] G. D. Hachtel and F. Somenzi, *Logic Synthesis and Verification Algorithms*. Boston, MA: Kluwer, 1996.
- [2] J. Dworak, J. Wicker, S. Lee, M.R. Grimaila, K.M. Butler, B. Stewart, L.C.Wang and M.R. Mercer, "Defect-oriented testing and defective part level prediction for commercial sub-micron ICs", *IEEE Design and Test of Computers*, Jan.-Feb. 2001, pp. 31-41
- [3] S. Ma, P. France, and E.J. McCluskey, "An Experimental Chip to Evaluate Test Techniques: Experimental Results," *Proc. International Test Conference*, pp. 663-672, 1995.
- [4] J. Dworak, B. Cobb, J. Wingfield, M.R. Mercer, "Balanced excitation and its effect on the fortuitous detection of dynamic defects," *Proc. Design, Automation and Test in Europe*, pp. 1066-1081, 2004.
- [5] P. Goel, and B. C. Rosales, "Test generation and dynamic compaction of tests," in *Dig. Papers 1979 Test Conf.*, pp. 189-192, Oct. 1979.
- [6] Hamzaoglu, J.H. Patel, "Test set compaction algorithms for combinational circuits," *IEEE Trans on Computer-Aided Design of Integrated Circuits and Systems*, pp. 957-963, August 2000.
- [7] F. Brglez and H. Fujiwara, "A neutral netlist of 10 commercial benchmark circuits and a target translator in fortran," in *Proc. IEEE Int. Symp. On Circ. Syst. (ISCAS)*, June 1985.
- [8] S. Lee, B. Cobb, J. Dworak, M.R. Grimaila, M.R. Mercer, "A new ATPG algorithm to limit test set size and achieve multiple detections of all faults," *Proc. Design, Automation and Test in Europe*, pp. 94-99, 2002.

VITA

Personal Information:

Name: Jason Michael VanFickell

Contact Information:

3914 Midforest
Houston, TX 77068-2915
jasonv@aggienetwork.com

Education:

- B.S., Computer Engineering, Texas A&M University, May 2004

Honorary Society Memberships:

- Tau Beta Pi
- Eta Kappa Nu
- Phi Kappa Phi
- Phi Eta Sigma
- National Society of Collegiate Scholars

Activities:

- Texas A&M University Institute of Electrical and Electronic Engineers
IT/Technology Chair and Webmaster, Fall 2003 – Spring 2004
- Texas A&M University Student Engineers' Council
Magazine Chair, Fall 2002 – Fall 2003