Optimization of Selection of Test Sets for Circuits

Olga Kuznetsova
Brown University Summer Research Student
University of Maryland, College Park
olga@umd.edu

Abstract

Generating a test set based on knowledge of the criticalities of all the faults in a given circuit can yield good test coverage. Yet criticalities are costly to calculate. In this paper, a method is proposed of minimizing the number of calculations of criticalities to as much as 12.5% of the original number of criticality calculations while maintaining similar product quality. Additionally, the test coverage for stuck-at faults is shown to apply to coverage of bridge faults for the same circuit.

Introduction

Integrated circuits are used in every part of modern life. These small chips are manufactured across the world for a huge variety of applications. These chips are relied on, and assumed to not be faulty when acquired. Yet when one manufactures anything in large quantities, defects often occur. The important thing is to catch them. Ideally, when it comes to defects in circuits, every defect that could possibly occur in them would be targeted and tested deterministically. Sadly, this would not be efficient, as there are too many possible faults that could possibly occur. We could also try to apply all possible test patterns to exhaustively test circuits, but as the number of possible patterns grows exponentially, this is also highly impractical. It is more efficient, if a little less safe, to use only a small amount of the possible test patterns that would most likely best detect the largest amount of the possible defects. The primary focus of this study is to find ways of detecting critical defects, and defects that are not detected often while refraining from calculating criticalities (importance of finding the defect) at each point. The studies outlined below have yielded ways of reducing the number of calculations of criticalities, while maintaining good coverage of all the simulated faults.

Relevant Terms

Before delving further into the description, some terms frequently used will be defined. Faults are logical models of defects that predict how a defect will affect the functional behavior of the circuit. For example, stuck-at faults are points where the circuit is fixed at either 1 or 0, regardless of the input value to that point. A bridge fault is a short between a group of signals (in this paper, two wires). A bridge fault can either be an OR bridge ('ORing' the two values of the wires it connects for both of them), or and AND bridge (same but with 'ANDing'). Faults should not be confused with ‘defects’, which are actual problems in a real-life circuit. Faults are used to simulate defects, in order to better detect defects. Faults are inserted into working circuits to see how to best detect them, and then the tests used to detect such faults are in turn used to detect defects. A term often used in the paper is the 'criticality' of a fault. This tries to capture the impact of a potential error on the user if a fault is not detected. The higher the criticality, the more catastrophic or noticeable the error, and the more important it is to detect the fault.

Related Work

In the real world, circuits come with faults, which could be bridge, stuck-at, or something else, so when they are tested for, they are all tested for together. When finding ways of best finding such faults, simulations must be run in order to trace the best ways of finding random faults. Since it is hard to make a general algorithm for all possible circuits, a specific circuit, color converter is used in most of the work of the laboratory, although the algorithms designed would work on all circuits. This is a giant circuit for which the criticalities of all stuck-at faults are available, thereby allowing it to be easy to find how important the faults found are. In the real world, determination of criticality is a very costly operation, and it is best to minimize the number of points that criticality must be calculated for.

Previous work has shown that as a defect becomes harder to detect, the overlap that its detection has with other defects gets smaller. [Dwor 04]. It has also been shown that random patterns are inefficient for capturing critical faults, if no criticalities are initially known. [Dwor 07].

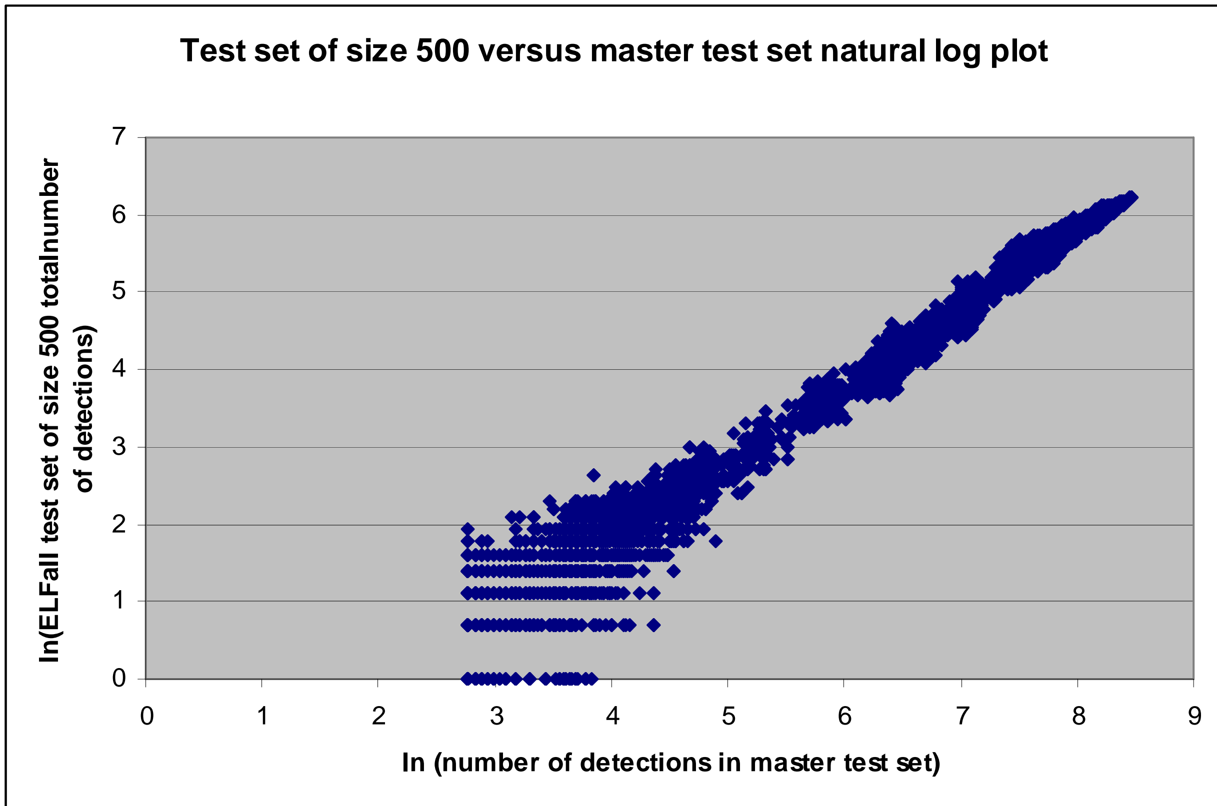
In a simple world, each fault would theoretically have a 50% chance of being discovered, as it can either have a value of 1, or 0. When in a complex circuit, a single fault is tested for, it is usually the case that many other faults randomly excited and therefore detected. This is called a ‘fortuitous detection’, one that was not planned for. Such detections are many, and add to the coverage of the circuit, but cannot be relied upon to cover all faults. It is best to have multiple detections of each stuck-at fault, yielding different methods of excitation of and larger coverage. A balance between purposeful excitation and random excitation needs to be achieved. [Dwor 04].

Previous work has also shown that, with some minor modifications, AND-type, OR-type an d4-way bridges are detected well by a good test-coverage vector-set for stuck-at faults. The edits made to the detection vector set for stuck-at faults when detecting bridges will not impact the previous detection of stuck-at faults. [Miya 05] A test that covers stuck-at faults can be modified to increase coverage of non-feedback bridges by 9% without disrupting the stuck-at fault coverage. [Miya 08].

This project aims to decrease the number of test sets and increase the test coverage of previous studies. The main study that is noted is that of Yiwen Shi, a graduate student in the laboratory. She created an algorithm for finding a list of the best vectors for each circuit based on the criticality values. She ranked vectors based on their 'ELF Value' which is defined as follows for each vector.

$$ELF_{all} = \sum_{i=0}^{n} \mathrm{d}_i * \mathrm{crit}_i$$

d is defined as either 1 or 0 depending on whether the given fault has been detected. n is the number of possible faults for the entire circuit. The vectors are then ranked from highest to lowest. Therefore, when looking for a test set of a certain size, the top 'X vectors are grabbed from the list. This is a very good approach, and is takes approximately 56% of the time of a full analysis. [Shi 08]



[Figure 1] ELF_{all} Top 500 Vectors versus Master Test Set detections

Figure 1 shows the graph of the benchmark vector list of size 500, versus the master test set (all vectors used). For the duration of this paper, graphs of natural logs of the number of detections will be used. This is a useful visualization because, ideally, there would be a 1-1 correlation between the number of detections of each fault for the old and the improved vector set. This ideal case would be represented by a slope of 1 in the graph, and an R^2 value of 1 (meaning densely packed data points).
Yet criticality is a very costly operation, as mentioned previously, so the goal of this study is to decrease the number of criticalities that have to be calculated while maintaining the high coverage of the vector sets. This study was used as the 'benchmark' for the current project. It is unknown whether the vector lists produced are the most efficient, but they are a good approximation of the 'best coverage' scenario. This study seeks to find a test set that will require less calculations of criticalities, and similar, if not better, test coverage.

Approach

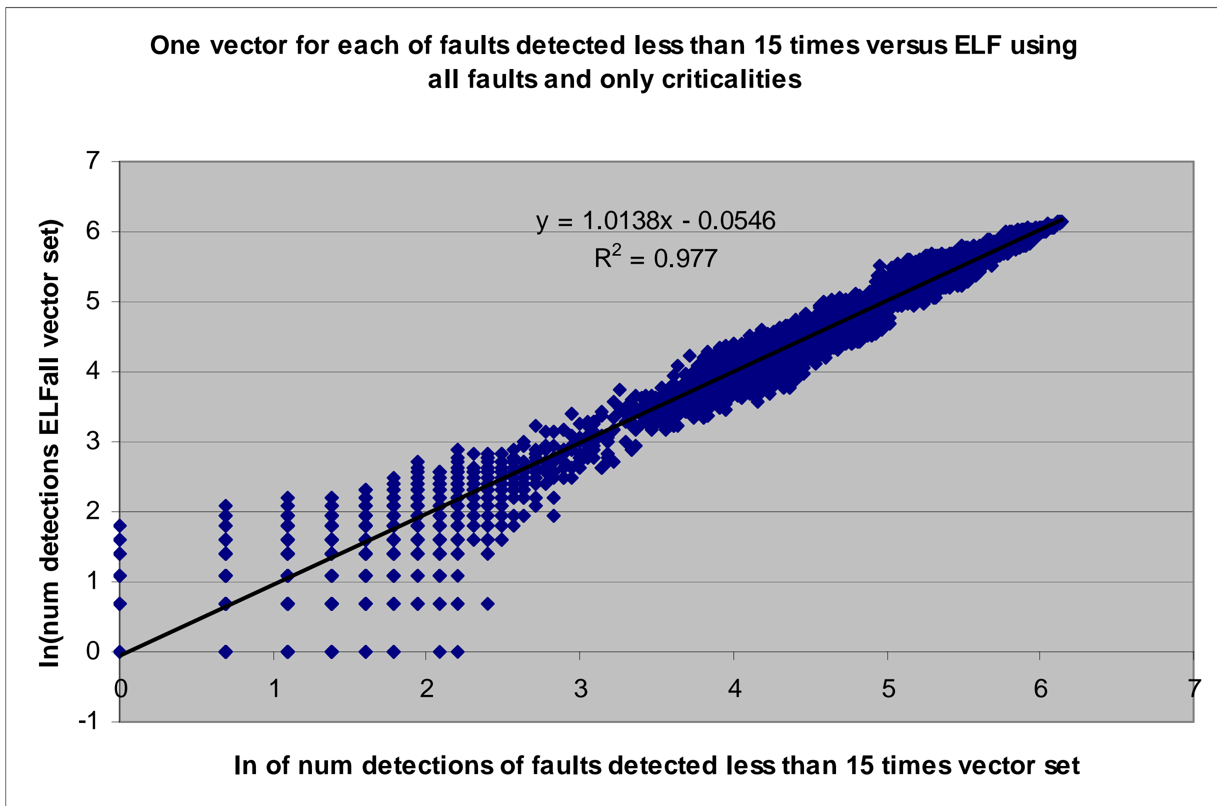
Fastscan

The main commercial tool that is used in the lab for analysis of circuits is FastScan by Mentor Graphics. FastScan is a program that allows one to load a circuit in and conduct simulations on it. Different types of faults can be added, and statistics can be generated about their detection. As FastScan is a commercial tool, it does have its limitations, forcing the data to be analyzed before and after the actual run resulting in a lot of pre- and post-processing. Another limitation is the amount of time it takes to run any FastScan program—any script needs to be tested on a smaller circuit before embarking on a 'colorconverter

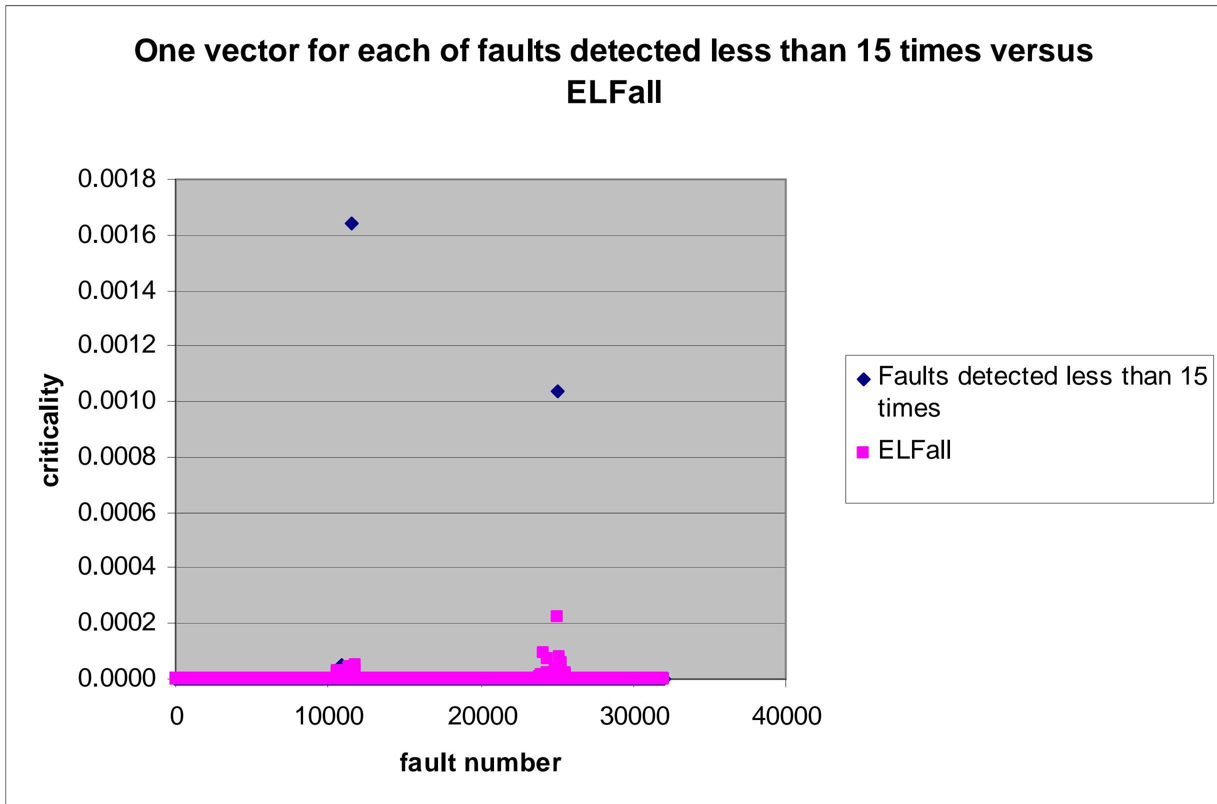
run'. FastScan is also poorly documented resulting in difficulties in having to re-program tasks that the program could probably accomplish.

Preliminary Approaches

The first approach taken to calculating the weight of the vectors was to give more weight to those vectors that detect vectors that are detected a small number of times. This was generally done by taking each fault that was detected less than a certain amount of times, and finding a vector that detected it, adding that vector to the list.



[Figure 2] Vector set of 464 vectors and their performance compared to benchmark



[Figure 3] Criticalities of faults not detected for test set size 464

As can be noted in the above two graphs, the correlation with the benchmark is quite good for this data set. This data set involved 464 vectors where each vector detected at least one fault that had less than or equal to 15 detections from the master test set. There are some faults with non-negligible criticalities that are not detected by the new vector set, even through in general it has comparable detections of each fault as the benchmark set.
It was putting too much stress upon small vector, which only detects a couple of faults, rather than grabbing first the vectors that detect the largest amounts of faults and are more efficient to run. This approach also ignored criticalities. It is interesting how a test that doesn't include any calculations of criticalities could produce such amiable results. This shouldn't be relied upon, as much of the detections in this case were due to 'fortuitous detection', rather than aiming to detect critical faults. It just happens that most faults that have high criticality also have many vectors that detect them for this circuit. This could not always be the case, and this should be provided for. Something different needed to be done.

Criticalities/ELF Values

The ELFall value that Shi used in her study was edited to create the vector set used for this study. The ELFall value was first stated to be, for each vector, the sum of all of the criticalities times the number of detections for each vector. The new ELF value includes a new variable: the number of detections. It combines concentration on less detected faults with concentrating on critical faults.

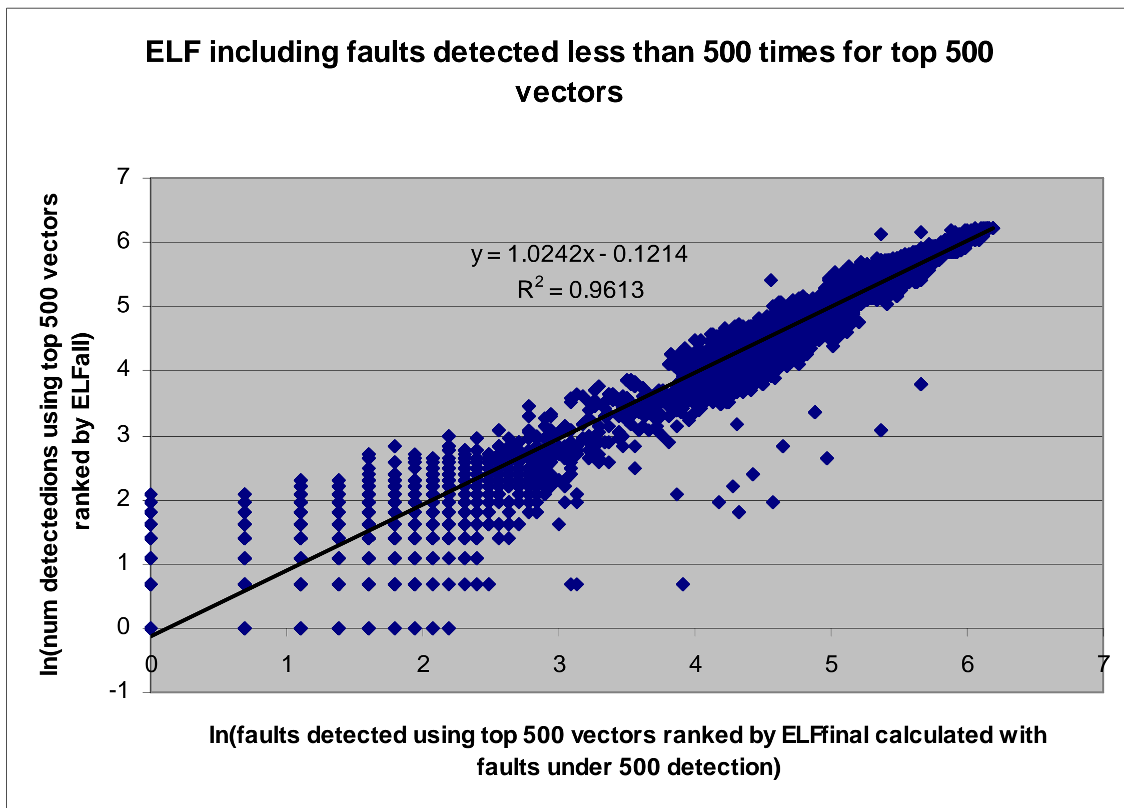
$$ELF_{new} = \sum_{i=0}^{n} \text{crit}_i * \text{num_detections}_i * d_i$$

Once again, d is either 1 or 0 depending on whether the fault is detected by the given vector. The vectors were ranked based on their ELFnew value same as before. A couple of problems were found with this approach. First, it still involved calculation of all of the criticality values, not improving on the number of calculations needed in previous study [Shi 08]. Second, it placed more value on the faults that have more detections, rather than those that are detected less.

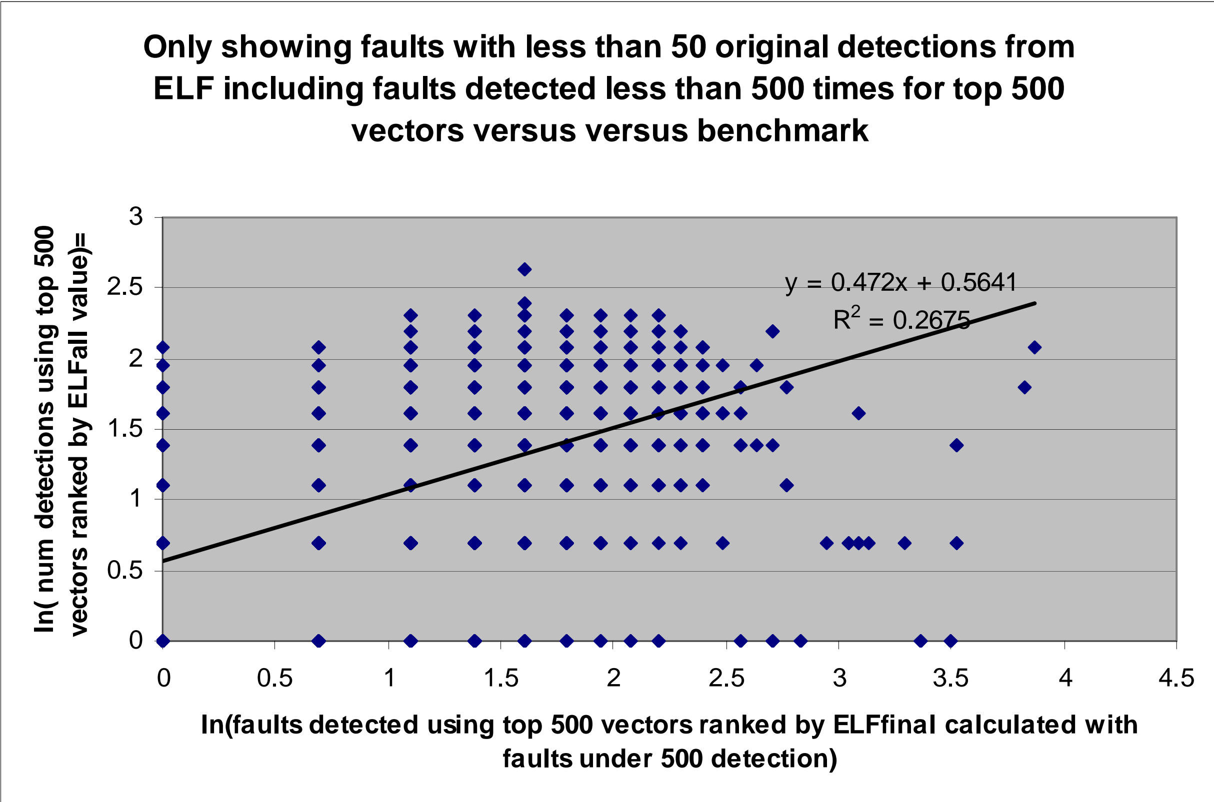
The solution to this problem was a modification of this formula.

$$ELF_{final} = \sum_{i=0}^{n} \text{d}_i * \text{crit}_i * (\text{MAX_DET} - \text{num_detections}_i)$$

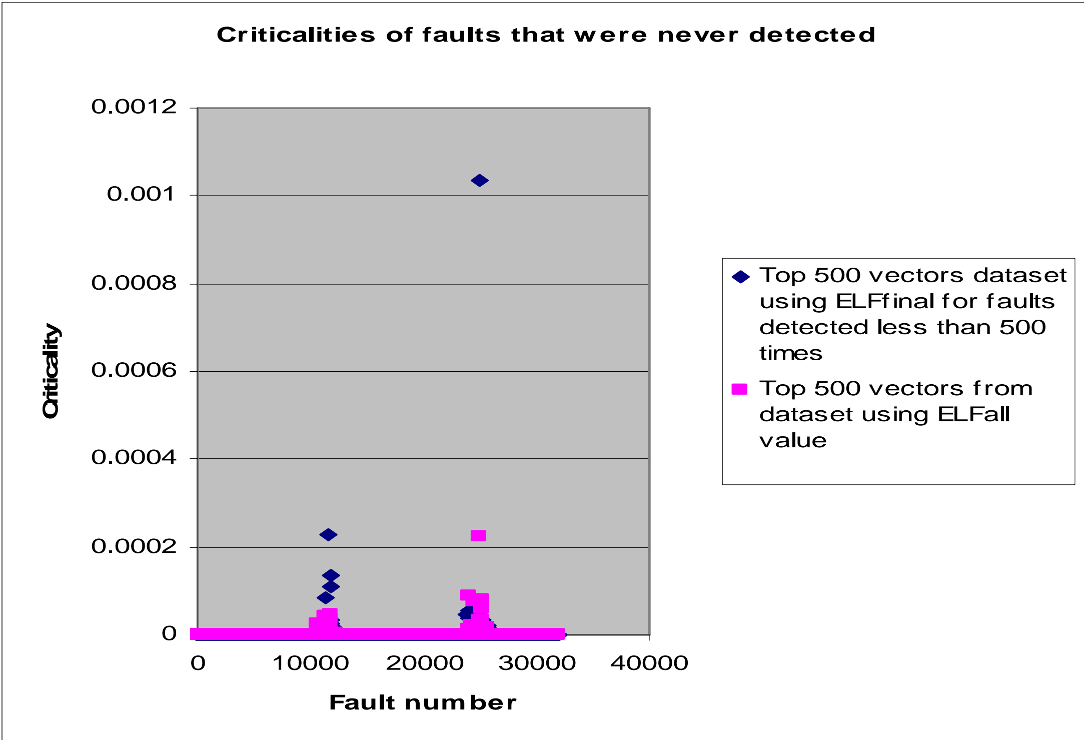
The only faults added in this equation are ones that have less than k total detection. MAX_DET is defined as the maximum number of detections of any fault for any vector in the set. The faults are sorted in order of number of detections for each vector to make it simpler. The results of this formula were taken for several values on 'k' as the 'best fit' for the new modifications. This also minimizes the number of faults that the criticalities have to be known for. These results then needed to be tested again to see how well they detected bridge faults. The following graphs represent a selection of comparisons between the benchmark vector sets of versus the new vector sets of the same length.



[Figure 4] Graph of natural log of previous data set versus the new data set



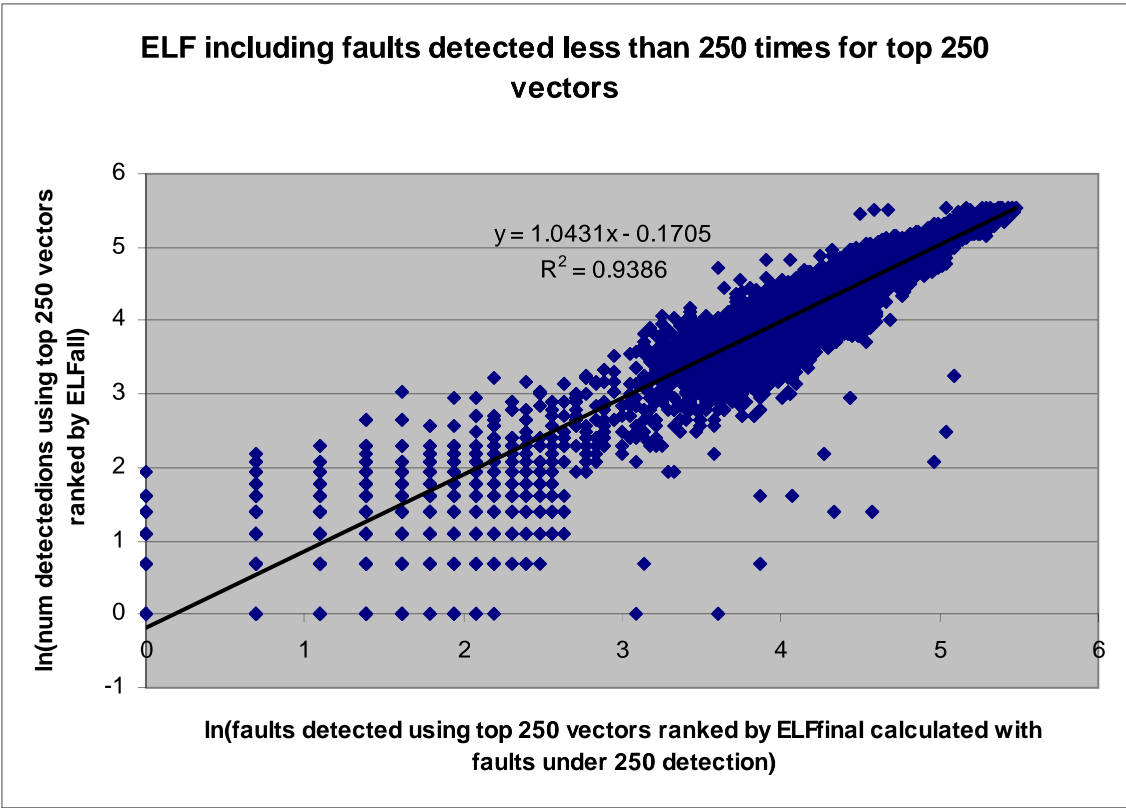
[Figure 5] Zoom around origin from Figure4



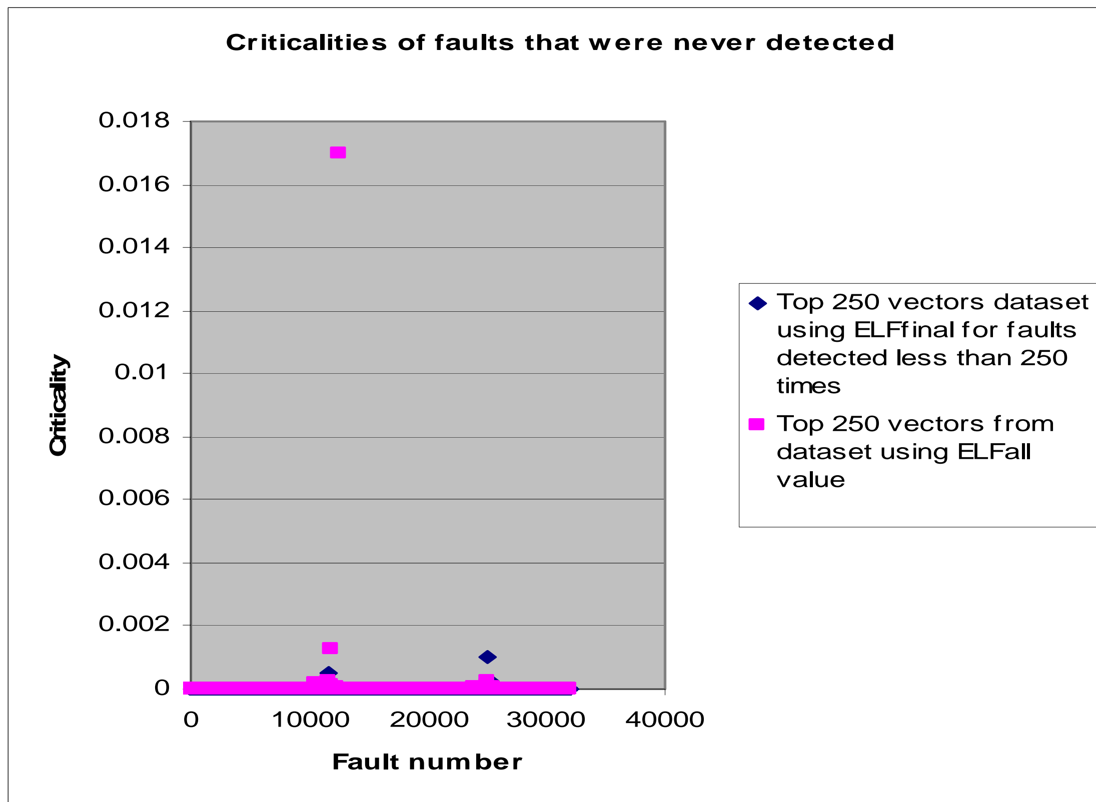
[Figure 6] Graph of only criticalities of faults not detected for dataset used if Figure 4

In the case of k = 500, 8354 faults' criticalities need to be calculated out of about 65000 which is about 12.5 %. A significant improvement. The correlation between the new vector set and the benchmark is good with a slope of 1.02. The R^2(correlation between points) value is close to 1, and so it appears like we have a good fit. The correlation between the two datasets is less favorable when examined only around the origin as in *Figure 5* with little if any correlation.

It can be seen in *Figure 6* that there are some faults that are not detected at all that have non-negligible criticalities, and that the previous data set scores marginally better than the new one. It should be noted that these criticalities are very small compared to the maximum criticality of 3.298 and the mean criticality of .101 for all faults compared to the highest point of the graph being 0.001557. In the previous studies, faults with criticalities less than .001 were neglected, so this graph would only show one undetected fault by the new test set—still a significant step.



[Figure 7] Top 250 vectors from ELFfinal using faults with less than 250 detections against benchmark



[Figure 8] Criticalities of faults not detected for Figure 7's data

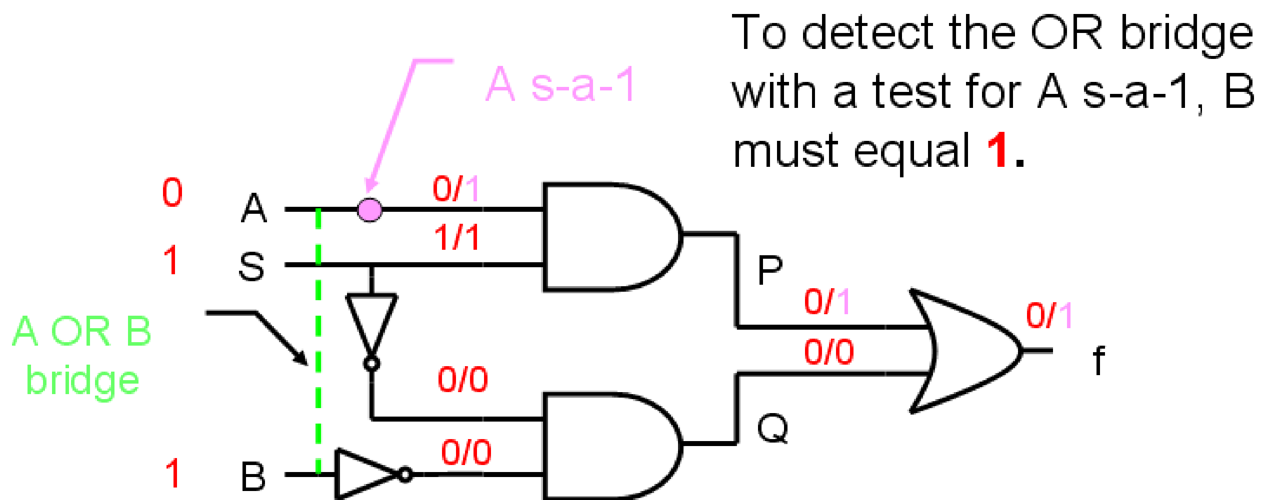
Figures 7 and 8 show another example of a different combination for calculating ELFfinal vector sets. The results produced are similar to those of Figure 4-6. It is not specified which combination of vector set size and number of detections to include in ELFfinal calculations yields the optimal result, as this could different for different circuits.

Bridges

In order to see if the vector set found through examining stuck-at faults and their criticalities is indeed good at catching untargeted defects, bridge faults needed to be examined as surrogates for untargeted defects. The reason for this is that bridge-faults are a very likely occurrence in an actual circuit, but most test generation tools do not explicitly model and generate tests for them. Thus, they must be detected fortuitously. For this reason, the same vector set that detects the stuck-at faults needs to be able to detect bridges. Stuck-at faults are more straightforward than bridges; with stuck-at faults every site of the circuit may be stuck at either 1 or 0. Bridges, on the other hand, involve multiple sites. The number of possible bridges grows as n^2 where n is the total number of sites. This is generally too large to simulate quickly, especially for a large circuit. Thus, we choose a random sample of all possible bridges. However, for our simulations, we also need to verify that the sampled bridges are non-feedback, so they do not cause oscillations. In addition, feedback bridges have previously been shown to be relatively easy to detect. Thus, we are less concerned that they will become test escapes. We also need to ensure that our chosen bridges cover a range of the circuit so that the circuit can be well represented.

The first approach I took to finding the bridges was to take every possible combination of

wires inside the circuit and connect them through the FastScan bridge fault option. It took long enough to generate that file, and it turned out to be about 10 GB in size—completely unloadable and unreasonable. I attempted to load this in pieces before realizing that not all the faults are non-feedback. Then came the question of actually finding the non-feedback edges.
The solution I employed was modifying a C++ program that finds all wires within a certain distance of each other. By definition, a non-feedback bridge would occur between a set of wires that are not within each other's input/output logic cones. In other words, the distance between them would be infinite because in the good circuit, they are in no way connected. The script was run to find all wires within any non-infinite distance of each other, and then for each wire, to grab all of the wires that are not within each others' list of connecting wires. There are some limitations to this approach as bridge faults generally occur between wires that are close to each other physically in the circuit. This approach generally finds wires on opposite ends of the circuit that would less likely have a short between them than those that are adjacent. Either way, the list of wire combinations produced is too large, so only a fraction of the bridges can be grabbed. I grabbed the first several combinations found for each wire, which is a rather simple approach, and could be improved upon in the future.
Once the bridges were acquired and formatted, they were combined with previous FastScan and stuck-at-fault data. Each of the wires of the circuit were mapped to the pins that they connect, and thereby the faults they are detected by. FastScan was run to find the value at each of the stuck-at-fault sites during the run of a good simulation with all of the vectors that could possibly be used for testing.



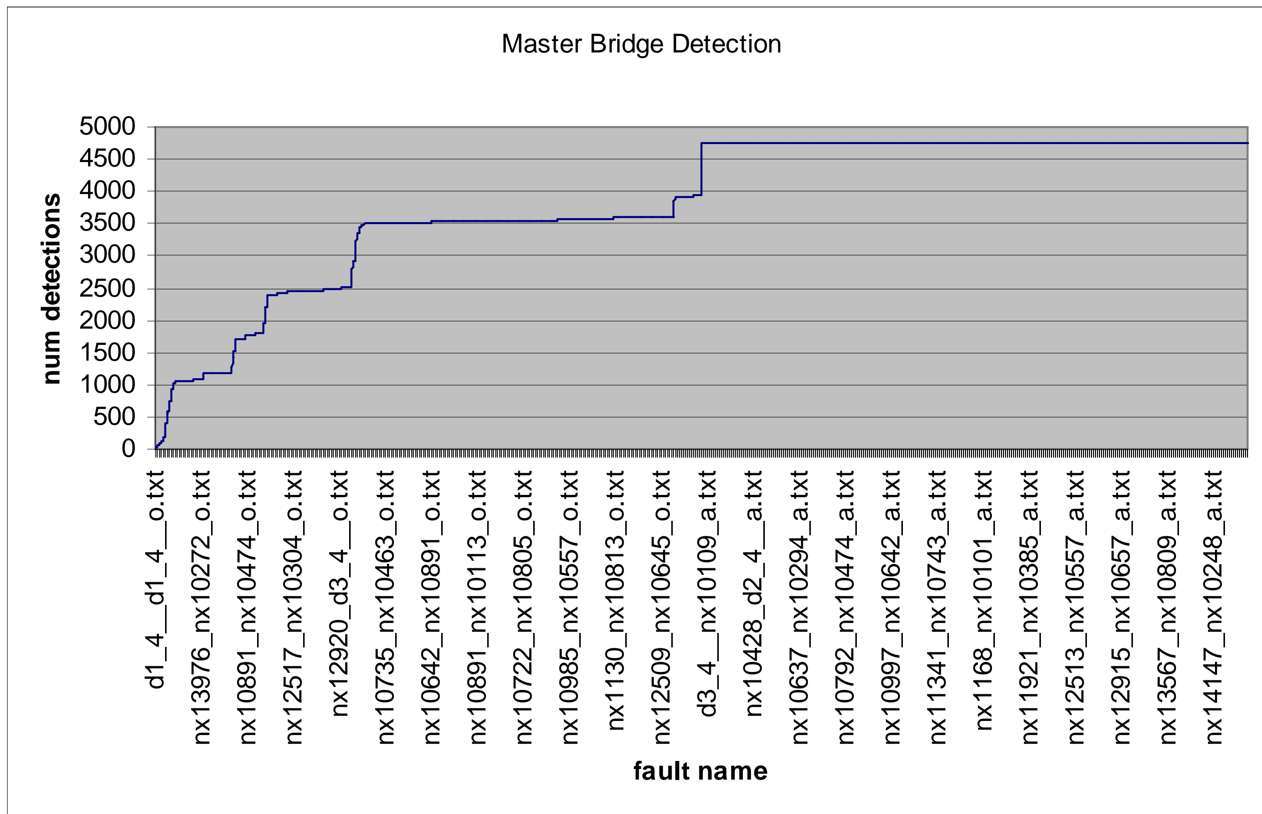
[Figure 9] Explanation of detection of OR bridge by knowledge of detection of stuck-at faults for the given area.

Figure 9 demonstrates how an OR bridge would be detected. The OR bridge (marked in green) would be detected if it is detected that A is stuck at 1, B must be equal to 1. If it is known whether a certain vector detects each type of stuck-at-fault, and what the value is at each wire during a 'good simulation' run of each of the vectors, it can be known whether any bridge would be detected.

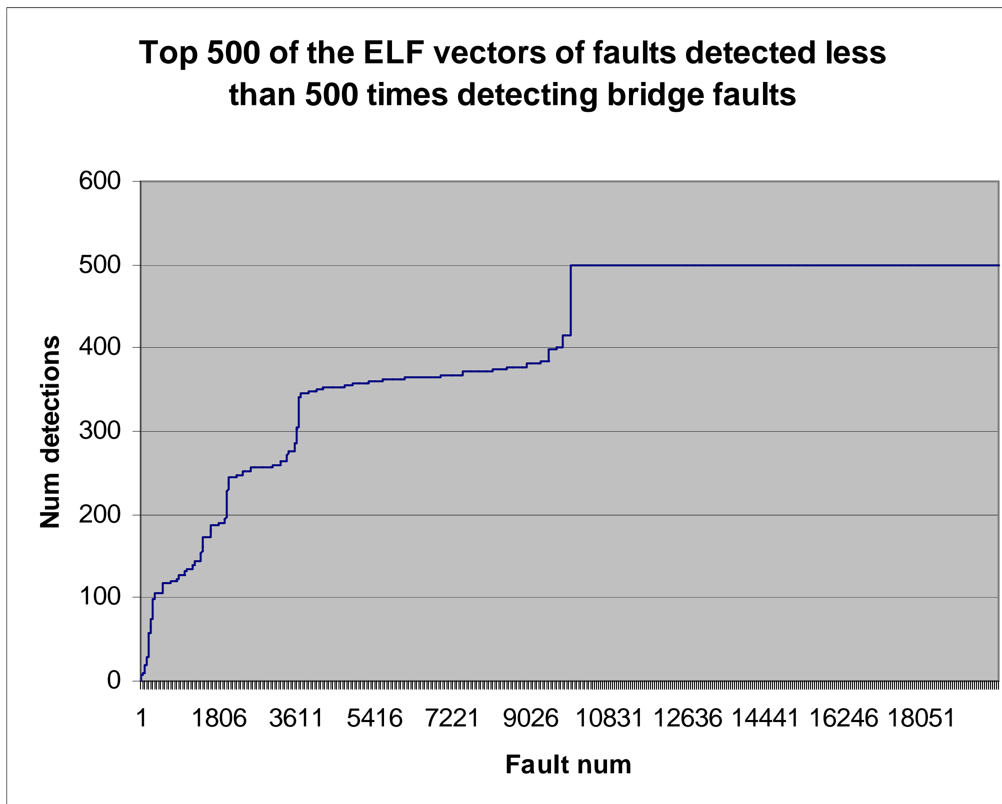
The bridges found were simulated by the following general algorithm:

foreach vector
foreach bridge from a --> b
// and bridge
if a is stuck at 1 and b is 0 at that point and bridge is detected
if b is stuck at 1 and a is 0 at that point and bridge is detected
// or bridge
if b is stuck at 1 and a is 1 at that point or bridge is detected
if a is stuck at 1 and b is 1 at that point or bridge is detected

This script produces a file for each of the possible bridges, and whether or not each vector detects them. From there, the vector list that was generated from the stuck-at faults can be used to find out how well the bridge faults were detected using the vectors that best detect stuck-at-faults.



[Figure 10] Master Graph for Bridge Detection



[Figure 11] Bridge detection using top 500 of the less than 500 detections ELF vector set

As seen in *Figure 10*, the ELF vectors detect bridge faults very well. There is only two bridges that are not detected, one of which is undetectable (is not detected even by the master test set). Similar results were yielded by using different ELFfinal vector sets, though it was found that in general, the more faults are included in the calculation of the ELFfinal value, the less bridge faults are missed.

Ideas for Future Work

Since it was not possible to conduct the simulation with all possible non-feedback bridges, only a small number of bridges were used to create the analysis. The bridges used were grabbed from the list of the first bridges found, and were not randomized. It is possible that the result would be different if a more random bridge-selection took place, but the location of the bridges required times that was not available in the current study. It would also be good to find the bridges that are within a certain distance, physically, of each other in the circuit. They would still be non-feedback, but there would be higher probability of them occurring as the wires could actually short.

One limitation of the stuck-at fault modeling is that many of the faults are looked at twice. The way that they faults were first assigned was to make each input and each output of each pin be stuck at 0 once at stuck at 1 once. This causes many wires to be reported as separate faults twice. This causes a bias in results, as they are counted twice in the analysis, compared to starting and ending wires (that go from only one point) are only counted once. It would be best to re-do the tests on a new set of stuck-at-faults that do not have repetitions.

It would also be beneficial to look at the particular stuck-out faults, and how they correspond to the bridges that are detected and not detected by any particular vector. The bridges that

are not detected could also be added to the FastScan run, manually, and it could be seen how much of an impact they make on other parts of the circuit. This would yield criticality numbers for the bridge faults, and could prompt a similar analysis of the bridge faults and vectors as was conducted with the stuck-at faults.
The preliminary assumption of this study was that the Shi 2008 vector set was the optimal pattern list for the detection of faults. It has been shown to be very similar to the new ELFfinal pattern list, but is different. It is also very close to the pattern list generated by only using number of detections (and no criticalities). It is important for future studies to know which way of determining pattern lists is more accurate. Since the Shi 2008 study only uses criticalities, and the new study uses both criticalities and number of detections, it would be beneficial to know what is more important in determining the usefulness of a vector: number of detections or criticality.

Conclusion

This study provided a wonderful improvement upon the Shi 2008 study. It showed that the ELF calculation, which was proved to provide a good way of ranking test patterns, can be improved by minimizing the number of criticalities that need to be calculated for each circuit. In the main example used for this study, the number of criticalities needed to be calculated was decreased to 12.5% of the original. The vector sets generated were also shown to be a good detection set for bridge faults, as a model for the detection of un-programmable possible faults. There are many ways that this study can be approached in the future, as outlined in the previous section, but for now it has shown that the direction that the Shi 2008 study went is fortuitous but could also be improved upon.

Acknowledgments

This research was performed at Brown University in Rhode Island as part of the Integrated Circuit and Computing Laboratory in the Electrical Engineering Department. Research was conducted under Professor Jennifer Dworak. Special thanks to the other members of the lab: Yiwen Shi, Elif Alpaslan, Stella Hu, Monica Noring, and Jom Kantapon. The author was sponsored by the Committee on the Status of Women in Computing Research (CRA-W) through their Distributed Mentor (DMP) program.

References

[Dwor 04] J. Dworak, B. Cobb, J. Wingfield, M. R. Mercer, “Balanced Exitation and Its Effect on the Fortuitous Detection of Dynamic Defects,” p. 21066, *Design, Automation and Test in Europe Conference and Exhibition Volume II (DATE’04)*, 2004.

[Dwor 07] J. Dworak “Which Defects Are Most Critical? Optimizing Test Sets to Minimize Failures due to Test Escapes,” *Proceedings of the 2007 IEEE International Test Symposium (ITC ’07),* Santa Clara, California, October 23-25, 2007.

[Miya 05] K. Miyase, K. Terashima, S. Kajihara, X. Wen, S.M. Reddy, "On Improving Defect Coverage of Stuck-at Fault Tests," in *Proc. of Asian Test Symposium*, pp.216–223, Dec. 2005.

[Miya 08] K. Miyase, K. Terashima, S. Kajihara, X. Wen, S.M. Reddy, “On Detection of Bridge Defects with Stuck-At Tests,” *IEICE Trans. Inf. & Sysm. Vols. E91-D. NO.3 March 2008.*

p683-689. 2008.

[Shi 08] Y. Shi, DiPalma Kellie, J. Dworak. “Efficient Determination of Fault Criticality for Manufacturing Test Set Optimization” *accepted for publication at the 23rd IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, DFT 2008.*, Oct. 1-3, 2008.