

February 25, 2009

CSCE 932, spring 2009: Home Work 3

(Test Data Compression)

Due: March 10, 2009

Test data compression is important in external (ATE-based) testing to reduce the cost of high speed storage on the tester. The purpose of this assignment is to explore the performance of various test-data compression methods using Hope and Atalanta as the test tools and ISCAS-89 scan circuits as the benchmarks. Then try both static and dynamic data-compression methods, as described below:

Static Compression: The basic approach you will follow is as follows:

1. Generate test vector for each fault in the collapsed fault list without random-filling of X values. This will give you the uncompressed data that has the maximum degree of freedom available for data compression (by allowing you to manipulate X's in any way you wish).
2. Apply each test data method on your list for comparison of performance to the above test data and collect performance data.
3. Summarize results, along with your analysis and interpretation.

I would like you to try at least the following test-data compression methods:

1. An image-compression based approach in which the collection of ternary (0, 1, and X) test patterns is viewed as an image for data compression, where each pixel in the image can have three values. Notice for scan designs, the rows and columns of the image can be permuted to improve the achievable compression. Use the best image compression method available to you.
2. Again, use an image-based approach but intelligently convert X's to binary values to provide maximum compression of the resulting binary image.
3. The scan-slice encoding method discussed in the class.

Dynamic Compression: Here the test generator and fault simulator are used *during* the process of data compression. An example would be as follows:

For each test-pattern generation step, after the test generator has produced a test for an undetected fault without X-filling, do fault simulation both to prune the fault list and to determine which as-yet-undetected faults are still candidates for detection by setting one or more X values to binary. Repeat this process until all X bits are filled or the list of detectable faults becomes null.

Illinois Scan is another example of a dynamic compaction method.

Look at the published papers in the area for how you might report the results of your investigation.