

A Pattern Skipping Method for Weighted Random Pattern Testing

by

Bernd Könemann

c/o IBM Corp. B56/052, P.O. Box 950, Poughkeepsie, NY 12602, USA

Abstract

Conventional Stored Pattern testing of very large scan design networks is increasingly burdened by the immense volume of pattern data that must be explicitly stored and manipulated for testing. Weighted Random Pattern testing uses an effective test data coding scheme to reduce the required explicit storage at the expense of increased test time.

In this paper, a novel method is introduced for efficiently skipping over ineffective WRP tests without adding any substantial storage overhead. Examples indicate that the method in a very practical way combines the short test times of Stored Pattern methods with the storage efficiency of WRP.

1. Introduction

Test data volume and test application time are always of concern for the cost effective manufacturing of complex microelectronic chips and packages. Leading edge technologies perennially stretch the capacity limits of state of the art test equipment. The test application problems, data volume and test time, are complemented by the difficulties of generating comprehensive tests for the ever more complex products.

Scan design techniques, like Level Sensitive Scan Design (LSSD) [1], have offered a solution for the test generation problem through automatic generation of high coverage tests within reasonable amounts of computer time. Many years of experience with scan testing have created a wealth of information about the relationships between the size of a network (i.e., gate count), expected test data volume, and test time. From this it is possible to extrapolate reasonably well how much scan testing will "cost" for future, denser technologies.

Scan design networks are characterized by a test mode in which the internal register elements are connected into one or more serial shift registers called scan paths. The scan paths are used to serially load test stimulus data into the register elements (load operation), and after some test

events have been applied, to serially shift out the test responses that have been captured into the register elements (unload operation). A single scan test, thus, can be a quite lengthy affair consisting of the load operation, a few test events, and the unload operation. The load and unload operations can be overlapped: while the responses from one test are shifted out of the scan paths on one side, the stimulus data for the next test are shifted in from the other side.

For large networks, the number of register bits in the scan paths by far exceeds the number of Primary Inputs (PIs) and Primary Outputs (POs), and loading/unloading the scan paths consumes most of the test time and data volume (stimulus and response data must be provided for each scan path bit). The test time can be reduced by implementing several scan paths in the network, making each individual scan path shorter. This shortens the load/unload time. It does not, however, improve the data volume because repartitioning into more scan paths does not reduce the overall number of register bits.

Leading edge technology has always stretched the limits of the available test equipment, but by continually upgrading the equipment and extending the test data buffers, the industry has been able to hang on by the skin of the teeth. There is a price to be paid for this: continually re-tooling and improving the test hardware becomes ever more expensive. The complexity, speed and pin count of chips and packages steadily increase, and each new tester generation has to have more and better pin electronics. Consequently, the price tag for staying ahead of the technology wave is steadily rising, and the test engineering community has continually been looking for ways to escape from this sky-bound cost spiral.

Some other factors have added to the urgency of change, at least for IBM. One factor is the desire to balance chip mix and production volumes with demand during the product proto-typing phase. In that time frame, small volumes of a very particular part type mix are required, which are best met by putting multiple part types (part numbers in IBM parlance) on the same wafer. To effec-

tively test such wafers, all respective part number test programs should fit into the tester buffer; otherwise a very expensive overhead for reloading the tester buffer will be encountered. Another factor is the recent addition of delay testing to the traditional DC stuck-at fault testing [2]. This substantially increases the number of tests and, thereby, puts an additional burden on test data volume and test application times. Finally, looking ahead into the near future, chips with more than 2000 pads (signal and power) are just around the corner. Wafer probing technology becomes a major challenge beyond 2000 pads; just imagine the mechanical force required for reliable electrical probe contact.

These trends became very visible in the second half of the 1980s, when highly complex CMOS chips and Multi Chip Modules (MCMs) started pushing the limits of existing equipment and methodologies. A study [3] of the trends and an analysis of known test technology options lead to the conclusion that the sky-bound test equipment cost spiral could be escaped by adopting a new strategy based on

- Boundary Scan and Reduced Pin Count Testing (RPCT), and
- Weighted Random Patterns (WRP).

Reduced Pin Count Testing uses Boundary Scan to largely relieve the external test equipment from the task of having to provide stimulus data on PIs or measures on POs. Each chip pin, except a limited number of clock and control pins, is "backed" by a Boundary Scan Cell. In one test mode, the internal logic is controllable from the Boundary Scan Cells on the input side and observable at the Boundary Scan Cells at the output side. Thus, no support from the external tester is needed on those pins for the purpose of testing the internal logic. This leaves the drivers and receivers still to be tested in a second test mode in which the receivers are observable at the respective Boundary Scan Cells on the input side, while the drivers are controlled from Boundary Scan Cells on the output side. By virtue of Boundary Scan, these tests are very simple and it is sufficient to provide simple DC parametric drive/receive capabilities on the attached external tester channels. Only a small subset of clock and control pins, including the scan channel inputs and outputs, need fully configured tester channels with deep pattern buffers. The resulting testers are substantially cheaper than testers with fully configured electronics for all pins. The current family of IBM's reduced pin count testers has 64 full function pins and a more or less open-ended number of reduced function pins.

Weighted Random Pattern testing uses an efficient test data encoding scheme to reduce the amount of explicit data storage required for the test stimulus data [4]. This is combined with Signature Analysis to eliminate the need for explicitly storing the test response data, for a total reduction of more than an order of magnitude. There is a drawback, though: when expanding the encoded tests on the tester, more tests are generated than would be needed for a stored pattern test with comparable test coverage: most of the expanded test WRP patterns do not detect any new model faults and, thus, do not contribute to the test coverage. As a result, WRP testing can take significantly longer than stored pattern testing.

The original study leading to the recommendation of WRP was conducted on the basis of DC stuck-at fault testing economics. On this basis, WRP was shown to be cost effective. Moreover, adding more tests gives WRP an edge in detecting more unmodelled faults, leading to better product quality. This was confirmed by fault simulation results and hardware experience [5]. Around the same time, an effort was started to implement full transition fault testing capabilities, with an objective of 95%+ transition fault coverage on top of 99%+ stuck-at fault coverage. Engineering intuition and small scale experiments suggested that this new objective would roughly triple the number of stored pattern tests compared to a simple stuck-at fault test. WRP was expected to fare somewhat better, "only" doubling the number of tests, because of its inherently better accidental fault detection. Still, WRP transition fault tests would be measurably more expensive, and a review of the expected return on investment seemed in place.

The prime unmodelled fault candidates giving WRP stuck-at fault tests and advantage over stored pattern tests were believed to be delay faults and, possibly, CMOS stuck-open faults. With transition fault testing a large class of delay faults and CMOS stuck-open faults no longer require accidental detection. Thus, not only does transition fault testing take longer to begin with, but it also puts the practical value of the additional WRP tests in question.

This motivated the search for methods to efficiently remove potentially "ineffective" WRP tests (i.e., tests not detecting any model faults) at the tester without having to explicitly store the test stimulus data. Ideally, such a method would combine the test time advantages of stored pattern testing with the data storage advantages of Weighted Random Pattern testing.

2. WRP Based Testing of Scan Design Networks

The main interest of the work presented here is in cost-efficient test methods for very large scan design networks. Such networks have a large number of internal register elements which for testing purposes are connected into several scan paths, also referred to as scan Channels.

WRP testing requires special hardware in the test equipment, and Figure 1 depicts a WRP tester pin.

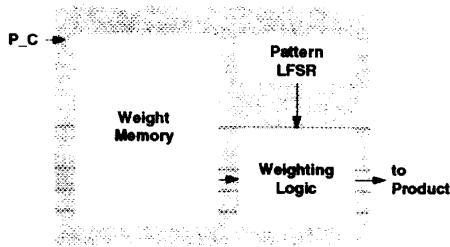


Figure 1: WRP Tester Pin

The WRP tester pin contains the Weight Memory, a pattern LFSR (Linear Feedback Shift Register), and some combinational Weighting Logic (see reference [4] for more details). The LFSR produces a sequence of Pseudo-Random Patterns under the control of a pattern generator clock, P_C. Each scan test requires to completely load the on-product scan Channels with new test stimulus values. The Channel load operation is accomplished over a number, L, of scan clock (S_C) cycles. L depends on how many shift register bits are in the longest Channel. For each S_C cycle a new scan-in value must be provided by the tester for loading into a particular Channel bit. To accomplish this, P_C and S_C are synchronized. Thus, a Channel Load operation involves L cycles of P_C and S_C. P_C also advances the Weight Memory address. It is assumed that each Channel bit position is associated with a particular Weight Memory address location which contains the weight values for this Channel bit. The Weight Memory address is reset to the same start value before each test pattern LFSR is continuously cycled without reset. Consequently, each Channel load operation cycles through the same weight value sequence, but the patterns generated by the pattern LFSR change between loads. The Weighting Logic, based on the weight values, biases the patterns to more suitable values for fault testing.

A single set of weight values for the SRLs and PIs is generally not sufficient for achieving high fault coverage. The WRP test generation software, therefore, generates a multiplicity of weight value sets. Each weight set is used for some number of tests. When to switch to a new set is

determined by the test generation software. For example, when a certain number of tests generated from a given weight value set fails to detect any previously untested model faults (or doesn't meet some other threshold criterion), the software will proceed to calculating and using the next weight value set. For details see reference [4].

A scan design network has other inputs besides the scan Channel inputs. The stimulus values at the vast majority of these additional pins are don't cares during the Channel load operation, and the WRP tester pins driving these inputs may be cycled along with those driving the scan Channels or may be left alone. If the test calls for updating the data PI values after the Channel load is completed, an additional P_C clock pulse is provided to update the pattern LFSRs, without pulsing the scan clocks. Clock pins and a few other special control pins receive repetitive sequences of deterministic values and are not driven with WRP values. The explicit tester storage required for the repetitive data is negligible.

A CMOS ASIC example with 300K logic circuits from [3], may be used to illustrate the test time impact of applying a full set of WRP tests. The example chip has 16 Channels with 1K bits per Channel (L=1000). The test application time is dominated by the time required to apply the 1K scan clock cycles needed in each test for loading the Channels. Assuming a 50ns scan cycle time, 20K Channel loads can be performed per second. To fully test the chip (99.9+% stuck-fault coverage) was estimated to take 27K stored pattern tests. At 20K loads per second, the tests can be applied in 1.35 sec. WRP testing for the same stuck-at fault test coverage was estimated to require somewhere between 10 and 25 times as many tests. That is, applying the WRP tests with comparable stuck-at fault coverage was expected to take between 13.5 and 33.75 seconds. Given the relatively low cost of the reduced pin count testers, these test times were considered tolerable.

Delay test changes the picture. Adding a 95%+ transition fault coverage objective was, as mentioned earlier, conjectured to roughly double the number of WRP tests, while tripling the corresponding number of stored pattern tests. The forecast for transition fault testing, hence, was $3 \times 1.35 \text{sec} = 4.05 \text{sec}$ for stored pattern tests, and between $2 \times 13.5 \text{sec} = 27 \text{sec}$ and $2 \times 33.75 \text{sec} = 67.5 \text{sec}$ for WRP tests. In a quality-driven environment the tests are applied under several operating corners (voltage, temperature, etc.). The time spent on weeding out the bad chips has also to be taken into account, and the test time spent per good chip, thus, is a multiple of the single pass test times stated above. Factoring all this in at typical equipment and engineering cost rates, leads to the conclusion that applying the full set of WRP transition fault tests

could carry a rather hefty price tag, even on reduced pin count test equipment.

3. Removing Ineffective WRP Patterns

The main concept behind reducing the WRP test times is to remove all or some of the WRP tests that do not add to the test coverage of stuck-at and/or transition faults. Candidates for removal are all tests that do not detect any model fault, called ineffective tests in the following.

WRP test time reduction, thus, requires to implement two additional functions in the WRP test software and hardware:

- In the test generation software a method to mark effective and ineffective tests, and
- In the tester hardware and software a method to skip over ineffective tests and only apply effective tests.

3.1 Finding and Marking Effective WRP Tests

The complete set of tests, effective and ineffective, is referred to as the full set of trial tests. The WRP software uses test generation methods to determine a set of suitable weight values and then generates groups of trial tests with these weight values. The group size is a function of the fault simulator (e.g., 256 for the algorithm in reference [4], 32 in the implementation referred to in [2]). Each trial test group is fault simulated to determine how many faults are detected by this group. Depending on the results, either another trial test group is generated with the same weight values, or the test generator is invoked again to find a new set of weight values. The process continues until all model faults are either detected or the test generator tried but failed to find a test (redundant or aborted).

The identification of effective WRP tests is performed in the fault simulation step. Assume, the WRP trial tests are numbered from 1 to nmax, spanning many test groups and multiple weight sets. To mark the effective and ineffective tests, the simulator simply keeps track of the currently simulated trial test number(s), and remembers the effective ones that detect one or more previously untested model faults. How this is best implemented depends, among other things, on the simulator type. For example, the Parallel Pattern Single Fault Propagate (PPSFP) algorithm used in [4] simulates 256 consecutive WRP trial tests in parallel. PPSFP is an event-driven fault simulator and stops fault propagation as soon as the fault effect reaches an observe point (network output or scan Channel bit). The fault is declared detected and any further pending fault effect activity is canceled (this is satisfactory for conventional WRP, where it is immaterial which

test(s) in the simulated group of 256 trial tests actually detected the fault: all tests in the group will be applied at the tester anyway). The simplest method for extracting effective tests from the group of simulated trial tests, is to take the simulation results at the state of detection, and identify and remember the number of the earliest test in the group which detected the fault.

This simple method may, however, not always be very efficient for producing a small, compact set of effective tests. Continuing to simulate might uncover a topologically longer fault propagation path that leads to detection in an earlier test number (but at a later simulation time). Further improvement can be achieved by storing not just the earliest, but all trial test numbers that detect the fault within the currently simulated group. This process is repeated for all remaining untested faults and the results are merged to achieve coverage of all detected faults with the least number of tests. Assume, for example, that two faults are detected by the current group. The first fault is detected by tests 17, 45, 176. The second fault is detected by tests 23, 45, 150, 201. Using the simple approach, tests 17 and 23 would both be marked as effective tests because each of them is the earliest test to detect a particular new fault. The more sophisticated algorithm would find that selecting test 45 as the effective test achieves the same coverage in a single test.

Some experiments was run to get a feeling for the "compactness" of extracted effective WRP. PPSFP was enhanced as described above and effective WRP stuck-at fault tests were extracted for 10 combinational chips from an IBM mainframe design. The results were compared with stored pattern test sets of equal stuck-at fault coverage. IBM's heavily used proprietary production level test generator had to be considered a good base for the comparison. Table 1 summarizes the results obtained in the experiments.

Chip	NPW	NPS	NPW/NPS
1	99	100	0.99
2	79	76	1.04
3	139	109	1.28
4	220	205	1.07
5	21	21	1.00
6	32	45	0.65
7	213	209	1.02
8	57	46	1.24
9	159	154	1.03
10	375	491	0.76
Sum	1394	1456	0.96

Table 1: Number of tests for WRP (NPW) and Stored Patterns (NPS)

The table lists the number (NPW) of effective tests marked in WRP and the number (NPS) of Stored Pattern

tests. NPW/NPS is the ratio of test numbers. The results suggested that test sets extracted from WRP and those from stored pattern test generation are similar in size.

In which form the information about the effective tests is best stored depends on what algorithm is used for applying the extracted test on the test equipment. The values that actually get loaded into the Channels for a given test are a function of the weight values and the starting state, S, of the pattern generator LFSR at the beginning of the Channel load operation. S is called the Seed for this particular Channel load operation. To get different test patterns without changing the weight values requires to change the Seed value, S, between tests. Each WRP trial test is, thus, characterized by an associated unique trial Seed, which is the state of the pattern generator LFSR has at the beginning of this trial test. The Seeds associated with effective and ineffective tests will be referred to as effective and ineffective Seeds respectively. The test numbers and/or Seeds can be used equivalently for representing the effective and ineffective tests.

3.2 Direct Re-Seeding Algorithms

Applying only effective WRP tests requires the ability to update the tester pin LFSR states to the Seed for the next effective test. This is equivalent to "skipping" over the Seed values of intermediate ineffective tests. The most obvious method is to directly store the effective Seed values in tester memory, and to load a new seed from this list at the beginning of each test (Direct Re-Seeding). This is storage efficient if the total number of Seed bits in all tester pin LFSRs is smaller than the total number of bits in a stored pattern test (assuming that both methods generate roughly the same number of tests). This can be illustrated for the CMOS ASIC example with 16 scan Channels of 1K bits each (i.e., L=1000). For simplicity it is assumed that in a reduced pin count test environment the Channel scan inputs absorb the bulk of the test data. The remaining inputs in the reduced pin interface are mostly clock and control pins with loopable data. A full deterministic transition fault test set was, as elaborated earlier, assumed to require 3*27K stored pattern tests (with 1 stimulus bit per test per Channel bit), or 2*200 WRP weight sets (with 4 bits per weight set per Input or Channel bit [4]). Allowing for some inefficiency in extraction, 3*30K effective WRP tests were considered reasonable. A stored pattern transition fault test then would take 3*27K*16*1K bits = 162M Bytes. If 32-bit tester pin LFSRs are used with direct Re-Seeding, each effective test is represented by a 32-bit Seed for each used tester pin LFSR. Thus, 3*30K*16*32 bits = 5.67M Bytes for the Seeds plus 400*16*1K*4 bits = 3.2M Bytes for the weight sets would be needed for direct Re-Seeding of

all tester pin LFSRs. The reduction is dramatic: roughly 9M Bytes for extracted WRP patterns with direct Re-Seeding versus 162M Bytes for stored patterns.

The single pass test application time for the direct Re-Seeding approach is estimated at 3*30K/20K sec = 4.5 sec, only slightly longer than the 4.05 sec estimated earlier for Stored Pattern testing.

3.3 Recursive Re-Seeding Algorithms

The direct Re-Seeding method described in the previous chapter still requires roughly 3 times the storage needed for full WRP. Better efficiency can be achieved by replacing the direct Re-Seeding method with a recursive approach.

Normally, the WRP pattern LFSRs are not explicitly Re-Seeded between tests. Instead, each LFSR is continuously cycled such that the final state from test n-1 becomes the Seed for test n (Implicit Re-Seeding). To advance to the effective Seed means to skip over intermediate ineffective Seeds. Assume, for example, that the next effective test to be applied after test n is test n+d+1 (that is d ineffective Seeds are to be skipped over) on an LFSR feeding a scan Channel of length L=1000. To advance the LFSR to the new Seed state would require to apply d*1000 cycles of the pattern generator clock P_C. This would finally leave the LFSR in the final state associated with test n+d, which is the Seed for test n+d+1. Doing so is tantamount to fully cycling through the d intermediate patterns, and no time is saved, unless the pattern generator LFSR can be cycled at a much faster rate than data can be scanned into the on-product Channel.

Cycling through all intermediate states is avoided by a more explicit Re-Seeding approach. Let S(n) be the Seed for trial test n. We define a Look-Ahead function LA, which is used at the beginning of each test to generate the new Seed from the previous Seed in a single clock cycle (Recursive Re-Seeding):

$$S(n) = LA(S(n-1))$$

This explicit Re-Seeding step is used in lieu of the implicit Re-Seeding used in conventional WRP test generation. The change from the implicit to the explicit recursive Re-Seeding is made in the tester hardware as well as in the test generation software.

LA should be easy to compute in software and need little extra hardware in the tester. In IBM's WRP testers, LA is implemented as a Linear Feedback Shift Register (Seed LFSR). The initial state of the Seed LFSR is chosen differently for each tester pin (e.g., randomly). The Seed LFSRs will then cycle through different Seed value se-

quences for each tester pin, which assures independence between the pattern LFSR sequences in different tester pins.

Figure 2 shows a simplified block diagram of a WRP tester pin with recursive Re-Seeding capabilities.

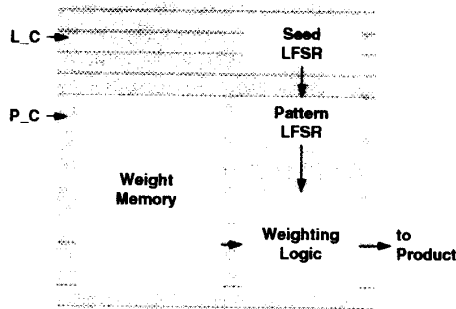


Figure 2: WRP Tester Pin with Explicit, Recursive Seed Generation

The explicit Seed value is stored in the Seed LFSR which is clocked by a Look-Ahead clock, L_C. Each L_C cycle will generate a new Seed value by cycling the LFSR. The WRP test generation software calculates how many ineffective trial Seed values are to be skipped over before applying the next test on the tester. For example, let the trial Seeds 1, 3, 4, and 12 be the only effective Seeds out of the first 16 trial Seeds submitted to fault simulation. The objective of Seed skipping on the tester is to only apply the tests corresponding to the 4 effective Seeds, while skipping over the ineffective ones. The sequence of the effective Seeds is characterized by the number, d, of ineffective trial Seeds to be skipped over before each effective Seed. Table 2 summarizes the resulting Seed-Skip data for this example:

Effective Test	Trial Seed Number	Skip Value d
1	1	0
2	3	1
3	4	0
4	12	7

Table 2: Example of Effective Seeds and Seed Skip Data

To skip over d ineffective trial Seeds during test application, the Seed LFSR is simply cycled d+1 times and the new Seed is loaded into the pattern LFSR. The maximum possible value of d depends on test generation parameters. The WRP test generation algorithm described in [4], for example, ensures that each group of 256 consecutive trial tests contains at least one effective test; hence, d cannot exceed 510 and 9 bits per effective test are suffi-

cient. Most actual values of d will be substantially smaller; the ratio between the number of WRP tests and deterministic tests was estimated to be on the order of 10:1 to 25:1. Assuming that the number of effective WRP tests is comparable to the number of deterministic tests suggests that the average value of d is on the order of 10 to 25.

The average 10 to 25 L_C cycles needed for Seed precalculation may appear at first glance, to add significantly to the test time. This is, however, not true for practical circuits. Test time is mainly a problem for large, complex circuits containing a large number storage elements in the on-product Channels. The test application time for these circuits is dominated by the time it takes to load/unload the Channels, that is the time required to apply the L cycles of S_C for the Channel load operation. For the example CMOS ASIC, L is 1000. The L_C cycles (less than 510, average of 10 to 25) for Seed precalculation could, therefore, be overlapped with the Channel load operation. As a result, we expect no measurable time overhead from precalculating the seeds except in very rare cases. Even if the seed skipping is not overlapped, it would on average add less than 3% to the test time.

The proposed method is very attractive from a storage point of view. Besides the initial Seed register states, no additional per-pin data are needed. To advance to the next effective Seed state the same number, d+1, of L_C clock cycles is applied to all tester pins. The values for d+1 can therefore be centrally stored, just 9 bits per effective test for the example algorithm. The Seed-Skip data for the extracted WRP transition fault tests of the CMOS ASIC example only need 3*30K*9 bits = 91.25K Bytes on top of the 3.2M Bytes storage for the WRP weight data. Hence, the Seed-Skip data storage is negligible compared to the WRP weight data storage. The hardware overhead for each tester channel is likewise very minimal (e.g., adding a Seed LFSR). The test time impact is essentially nil and the expected test time, therefore, is the same as that for direct Re-Seeding (4.5 sec for a single pass transition fault test of the example CMOS ASIC).

One nice side effect of the recursive approach is that the Seed skip values, d, are the same for all tester pins and only need to be stored once. The data overhead, thus, is independent of how many tester pins are used, and the method can be applied to full pin count testing without additional storage cost.

3.4 Combinational Re-Seeding Algorithms

Although the recursive method described in the previous chapter is simple and does not degrade the tester

throughput, some purists might prefer a more "combinational" Seed generation algorithm. The objective would be to always calculate the next effective Seed in one clock cycle. This requires to find a viable combinational function to directly map the effective test numbers into seeds for the tester pin LFSRs. Of some concern is to ensure that the resulting seeds do not lead to unwanted correlation between the tester pins.

The simplest way to derive sufficiently uncorrelated Seeds for the different tester pins is to use different "hash codes" to transform the common test number into a different Seed value for each tester pin. How much de-correlation is required for optimal performance is not easy to guess. Therefore, it is impossible to elaborate all possible algorithms that might be appropriate. It should suffice to say that such algorithms can be and have been conceived. However, we feel that the recursive method is quite adequate for our purposes and we have not pursued the combinational Re-Seeding methods in any depth.

4. Some Results and Discussion

The recursive Re-Seeding method has been implemented in the IBM WRP test generation software and some test equipment hardware. Initial results verifying the potential test time reduction are available. The guesstimates presented earlier in this paper suggested to expect a 10 to 25 times reduction in tests when extracting the effective tests from the full set of WRP trial tests. It is surprising that these rough estimates based on engineering speculation and some very limited experimentation are actually not far off reality. Table 3 shows some results obtained for a number of somewhat smaller CMOS ASIC chips.

Chip	Full WRP DC	Extr. WRP DC	Ratio DC	Full WRP AC	Extr. WRPAC	Ratio AC
1	81K	3.8K	21:1	138K	7.2K	19:1
2	66K	3.9K	17:1	125K	8.5K	15:1
3	34K	1.4K	24:1	40K	3.0K	13:1
4	39K	2.2K	18:1	-	-	-
5	62K	2.8K	22:1	-	-	-

Table 3: Results for Extracting Effective WRP Transition Fault Tests

The results clearly are not statistically representative, and the transition fault test generation system [2] is still too early in its life-cycle to be fully optimized. However, the results do validate that significant test time reduction is possible without sacrificing storage efficiency.

One open question is the impact on product quality. Removing nominally ineffective tests reduces the number of

applied tests and certainly limits the chance of detecting some unmodeled faults. How big that exposure really is remains to be seen. Even after extraction, the effective transition fault test set considerably larger than the traditional stuck-at fault test sets; and, they definitely are more effective for defects requiring pattern pairs (delay defects, CMOS stuck-open faults). Hence, one can speculate that the ineffective patterns will have little, if any, measurable effect on quality. The final answer will depend on hardware experiments.

The recursive Re-Seeding method has the advantage of extremely small storage overhead per used test. Some ineffective tests can be added without compromising the storage efficiency. The IBM implementation has user definable options to not discard all ineffective tests. The user can add as many ineffective tests as reasonable within a test time budget. Thus, a full continuum of trade-offs is possible, combining the best of WRP storage efficiency with stored pattern tester throughput.

5. Summary

We have presented the concept of using Weighted Random Pattern tests generated with LFSRs (or other finite state machines for generating Pseudo Random Patterns) to derive compact, coded data representations for effective fault tests. Each effective test requires a few bits on top of the WRP weight data to characterize which of the WRP trial tests are effective and, thus, should be applied at the tester. Several methods for extracting the effective tests in real time on a tester have been described. With the proposed methods it becomes possible to combine the test data reduction of WRP testing with the compactness and throughput of stored pattern methods. Based on our initial experience with a WRP test system for scan designs, we expect an overall test data reduction of up to two orders of magnitude compared with traditional methods. The new methods could be fruitful not only for manufacturing test, but also in service processor based system Self-Test applications.

6. Acknowledgments

The author gratefully acknowledges the invaluable contribution from John Waicukauski, now with CheckLogic, who implemented and performed the initial simulation experiments for extracting effective WRP tests.

The paper would not have seen the day of light at the ETC '93 without the initiative and support of my manager, Tom Williams. Thanks, Tom.

References

- [1] E. Eichelberger and T. Williams, "A Logic Structure for LSI Testability", Proc. 14th Design Autom. Conf., pp. 462-468, 1977.
- [2] B. Koenemann et al., "Delay Test: The Next Frontier for LSSD Test Systems", Proc. Intern. Test Conf., pp. 578-587, 1992.
- [3] R. W. Bassett et al., "Low-Cost Testing of High-Density Logic Components", IEEE Design & Test of Computers, pp. 15-28, April 1990.
- [4] J. Waicukauski, "A Method for Generating Weighted Random Test Patterns", IBM Journal of Research & Development, Vol. 33(2), pp.149-161, March 1989.
- [5] J. Waicukauski and E. Lindbloom, "Fault Detection Effectiveness of Weighted Random Patterns", Proceedings International Test Conference, pp. 245-255, 1988.