COMPUTING PRACTICES

Dynamic Scan: Driving Down the Cost of Test

Samitha Samaranayake
Nodari Sitchinava
Massachusetts Institute of Technology

Rohit Kapur
Minesh B. Amin
Thomas W. Williams
Synopsys Inc.

Imagine saving even fractions of a cent on each of thousands of test patterns per chip. Multiply that by the billions of chips fabricated each year, and you have a glimpse of the benefits dynamic scan could offer.

Dire predictions about the soaring cost of semiconductor test are all too familiar. Two factors primarily drive this cost: the number of test patterns applied to each chip and the time it takes to run each pattern. For example, typical semiconductor testing for each chip involves a set of as many as 1000 to 5000 test patterns—sets of input values and their associated (expected) output values. These test patterns are applied through scan chains that operate at about 25MHz. Depending on the size of the scan chains of the chip, a set of test patterns for a typical chip can take few seconds to execute per chip. It's easy to see that even a small decrease in either the number of patterns or the time to execute them can quickly add up to big savings across millions of fabricated chips.

This potential savings forms the basis for dynamic scan, a new approach to the well-established scan test methodology. Our initial studies, presented here, indicate that dynamic scan could easily

reduce the time taken to apply the test patterns by about 40 percent. A more theoretical analysis shows a potential savings of as much as 80 percent.

CURRENT SCAN TEST USAGE

Current design-for-test technology creates scan chains that automatic test pattern generation (ATPG) use as control and observation points. Basic scan technology attempts to maximize the number of scan elements, which supports the verification of design specifications such as timing.

These scan chains provide controllability and observability for the automatically generated test patterns created later in the design process.[1] All scan elements go into a set of possibly balanced chains to provide a single multichain scan configuration that automatic test pattern generators use in generating test patterns.

Prior research, however, has recognized that the maximal scan configuration is overkill for ATPG. ATPG doesn't need to scan all the memory elements to attain the required fault coverage. Instead, the *partial-scan* technique identifies a smaller set of memory elements, providing a single configuration that ATPG can use to attain the required fault coverage.

However, a single scan chain configuration restricts an ATPG-based method to using all the scan elements for all test patterns. Although the test must scan all memory elements at least once to detect the faults, *it need not scan every element for every test pattern*. A method that pares down the number of scan elements for each test pattern could provide substantial test application time and cost savings**.**

Past solutions have used subsections of a single-scan-chain architecture to apply tests to different design modules. For this strategy to be effective, these modules must be well-bounded and have independent test patterns,[2,3] characteristics not found in today's increasingly complex designs.

Our work expands previously defined concepts for single scan chains to provide a new architecture for use in conjunction with ATPG. We want to apply test patterns to random logic, but use the shortest possible scan chains. To do so, we blend the benefits of using multiple scan chains[4] with the reconfiguration method for single scan chains. These methods work together to reduce test data volume and application time.

We intend this technology for use with random-logic ATPG and its patterns. Thus, our solution avoids breaking a basic concept employed by today's scan-chain construction methods: Multiple scan chains that are active at any given time have a single path between the scan-ins and scan-outs of each scan chain. This distinguishes our solution apart from more radical solutions that fan out scan chains from a single scan-input.[5]

BASIS FOR DYNAMIC SCAN

Figure 1 shows a circuit that has one scan chain containing five scan flip-flops. It has scan-in signal SI, and scan-out signal SO. The scan enable SE signal configures the flip-flops for scan operation, while the clock Clk operates the scan chain.

Figure 1. Generalized circuit with a single-scan chain.

Table 1 shows the stimuli and response vectors (after compaction but before random fill) needed to detect the faults in the circuit of

Figure 1. The tester applies the stimuli at input a, and each of the five flip-flops, c_0 through c_4. It records the responses at outputs u and v, and each of the five flip-flops. The inputs or outputs are logic 0, logic 1, or don't-cares (denoted by X).

Table 1. Stimuli and responses necessary to detect design faults in the circuit of Figure 1.

Test patterns	Stimulus (a,c_0, …, c_4)	Response (u, v, c_0, ..., c_4)
T_1	1 X X 1 0 X	0 1 X X 1 X X
T_2	0 X X 0 1 1	1 X X X X X 1
T_3	X X X 1 0 0	0 1 0 X X 1 X
T_4	X X 1 1 0 1	1 X 1 X 0 X X

We apply these test patterns in a three-step sequence:

- scan-in T_n,– scan-out T_{n-1};
- stimulate inputs, measure outputs; and
- pulse a capture clock.

A tester first scans the data into the flip-flops, applies a stimulus to the inputs, and measures the circuit outputs. It then applies a pulse on the clock signals. The pulse triggers an update of the scan chain flip-flops and thus capture the design's response to the test pattern. The tester then scans out the response; it simultaneously scans in the next test pattern.

For the fixed configuration in Figure 1, the test patterns would operate the scan chain of length five. This scan operation dominates the test application time, taking five clock periods in the example scenario. Every test applying a stimulus to or measuring a response from the scan flip-flops would perform this scan operation and consume these five clock periods.

Each test pattern in our example operates the scan chain; total test time per pattern is 5 cycles for scan in of T_n and scan out of T_{n-1} plus 1 cycle for updating the flip-flops plus 5 cycles for the scan out operation of the last test pattern (that could not be overlapped with other tests). Running the entire test of four patterns consumes $(5 + 1)) \times 4 + 5 = 29$ cycles. Of this, the scan time is 25 clock cycles. The scan operation's duration is independent of the number of scan values the test needs. The rigid configurations requires that the tester scan every cell, so typical ATPG algorithms would randomly fill the don't cares and provide fully specified test patterns.

In our work it is recognized that the scan chain doesn't need to be the same for all tests. The scan chains should ideally provide access to the flip-flops that the tests need. Figure 2 shows the scan chain structure that would allow this access.

Figure 2. Scan chain structure that allows dynamically configurable access to flip-flops. This scan chain can mimic test patterns that the scan needs.

Signals that control the multiplexers let them either bypass or include a flip-flop in the scan chain. The multiplexer control signals can come from circuit inputs or from a control-register-like configuration. The configuration in Figure 2 can include or exclude any flip-flop from the scan chain, tailoring the scan chain to suit the test pattern.

Table 2 lists the example test results for Figure 2. A "-" signifies a

value that was not applied in the test pattern by using a configuration of the scan chains that does not include the associated flip-flop.

Table 2. Test stimuli and responses for the example in Figure 2.

Test pattern	Stimulus (a, c_0, ..., c_4)	Response (u, v, c_0, ..., c_4)
T_1	1 - - 1 0 -	01 - - 1 X X
T_2	0 - - 0 1 1	1X - - X X 1
T_3	X - - 1 0 0	01 0 X X 1 X
T_4	X X 1 1 0 1	1X 1 - 0 - -

Test T_1 uses scan cells c_2 and c_3. Tests T_2 and T_3 use scan cells c_2, c_3, and c_4. Test T4 uses c_0, c_1, c_2, c_3, andc_4. The total scan time for all test patterns is 2 + 3 + 3 + 5 + 2 = 15 cycles (because we take advantage of the overlapping scan ins and scan outs), which is much less than the total scan time of 25 cycles for the original scan chain.

This is a very expensive configuration in terms of supplying the multiplexer control signals. It would require many inputs control the multiplexers; using a control register, on the other hand, would require loading the register for every test pattern. Accounting for all these considerations might be impractical in an actual circuit layout.

A more realistic approach limits the number of control signals by making multiple patterns use a single configuration.

USING SCAN SEGMENTS

A practical implementation of dynamic scan must focus on ensuring the following:

- Fewer configurations mean fewer (expensive) multiplexer control signals, so a practical implementation should use a minimum number of configurations.
- An efficient implementation should order multiple test patterns to use a single configuration. This ordering should also maximize scan-in and scan-out overlap, and minimizing configuration setup overhead on a per-test-pattern basis. Patterns using the same configuration should run consecutively.
- Most test patterns should use short scan chain configurations.

So for dynamic scan to be useful, we must create segments to limit the number of supported configurations. *Segments* are contiguous scan chain components that a scan test must bypass or use as a set.

The benefits that dynamic scan could provide depend on segment identification. Numerous methods are available for identifying scan segments, including topological traversals of the design or the use of ATPG engines and test patterns to guide the selection process.

Figure 3 shows some example dynamic-scan configurations that use segments. These configurations account for the fact that all the patterns in our example set use the last three scan cells.

Figure 3. Dynamic scan configurations that permit bypassing of the first two scan cells by routing signal si *around these cells through an alternate path (a) or providing an alternate input,* si_2 *(b).*

Preventing test patterns from excluding individual scan cells offers a simpler solution than full-blown dynamic scan, but doesn't offer as

large a reduction in test data volume and application time. In our example, the scan segments force pattern T_1 to use the c_4 scan cell. In this case, ATPG can randomly fill the don't-care for c_4 in this test pattern. The dynamic scan chain implementations shown in Figure 3 provide an overall scan test application time (and proportional test data volume) of 3 + 3 + 3 + 5 + 5 = 19 cycles.

Any reasonable dynamic-scan partitioning includes several patterns that use each of the different configurations. If only one pattern needs a configuration, test engineers should revisit the segment identification process.

At one extreme of the dynamic-scan solution, a test can selectively bypass all scan elements. That is, segment length equals 1. This configuration, the most flexible one, provides maximum benefits at the expense of design-for-test and layout problems.

At the other extreme, a test does not bypass any scan cell, and the original scan chain is the only configuration available to the test patterns. This least-flexible configuration does not effectively reduce test data volume or test application time, but it has minimal additional impact on the typical scan chain layout problems. Our goal falls in between these two extremes: We seek to achieve significant benefits with a small number of segments.

DYNAMIC SCAN WITH MULTIPLE SCAN CHAINS

Although our examples use a single scan chain, applying this to all scan chains independently would create significant overhead problems. For this reason, the most promising concept in making dynamic scan a reality is the use of multiple scan chains within a dynamic scan segment.

To reduce overhead, a configuration must create multiple scan chain partitions. Each partition consists of parallel scan segments connected in series to segments of other partitions. Multiplexers, which allow activation or bypassing of all a partition's scan segments, separate the partitions.

Figure 4 shows a dynamic-scan design with multiple scan chains. ATPG patterns operate three parallel scan chains. The length of the three available scan chains depends on the control register values. For example, if 0 on the multiplexer control line selects the top input for all multiplexers, then 100 in the control register would give three short parallel chains consisting of the leftmost scan segments of partition 1. A 101 code would make three scan chains available, consisting of all the segments in partitions 1 and 3.

Figure 4. Multiple scan chains with dynamic scan. ATPG test patterns operate three parallel scan chains, the lengths of which depend on the control register values.

This implementation permits each active scan chain to incorporate one scan segment from each partition. Each partition's scan segments should be as balanced in length as possible to achieve maximum benefits in terms of test application time. For maximum benefits, each partition should have the same number of scan chains.

EXPERIMENTAL RESULTS

Dynamic scan can significantly reduce test data volume and test application time. These benefits rely on the fact that designs have constraints (circuit structure dependencies) and that ATPG and its associated compaction algorithms are imperfect. Dynamic scan relies on the presence of several don't-cares in test patterns, even after

ATPG compaction. So to assess dynamic scan's real-world potential, we determine the number of don't-cares in test patterns (after compaction) for realistic circuit designs.

Table 3 lists characteristics for the designs we used in our experiments. The design names include the number of primitive logic gates in this circuit, as perceived by the ATPG tool. The table lists the scan flip-flop count for these multichain scan designs. Although the table includes the fault count and fault coverage for the circuit's associated scan test, the most relevant number is the total number of patterns for the original scan test without the implementation of dynamic scan.

Table 3. Designs for dynamic scan analysis.

Design name	Number of scan flip-flops	Number of faults detected (1000s)	Number of test patterns	Fault coverage (percentage)
A: D118K	8,782	283	711	99.55
B: D87K	8,570	270	1,154	96.90
C: D90K	9,181	223	3,761	99.69
D: D198K	15,180	463	1,951	91.64
E: D296K	9,307	709	2,240	98.67
F: D259K	1,024	262	270	99.50
G: D37K	1,862	73	1,673	100.0
H: D44K	2,851	78	1,725	99.91

Typically, in any test set, the first few patterns detect many faults, and most subsequent test patterns detect very few. Thus the first few patterns are by nature fully specified (with very few don't cares), they typically use the complete set of scan flip-flops.

The remaining patterns are more likely targets for dynamic scan. Thus, for the following experiments, we created about 200 test patterns and forced them to use the complete scan chains. After performing fault simulation for these patterns, we let the ATPG tool perform all the compaction it could. We then counted the number of don't-cares in the test patterns.

Figure 5 shows that most test patterns have many don't-cares, thus an approach like dynamic scan could reduce the test data volume and test application time. This data also shows that the compacted ATPG tests are so sparsely populated that it should be possible to identify partitions that cover many tests.

//Author: You lost me in Figure 5—don't-cares as a percentage of what? All possible input values? And, what is the significance of "after random fill vectors"? << If you remember in the previous sections stimulus and measures can happen at inputs, outputs and scan-flip-flops. The don't cares are a percentage of the total. That is most test patterns have very few 1's and 0's and many don't cares. >>//

Figure 5. Percentage of don't-cares after a random fill of the test vectors. Because most test patterns have many don't-cares, dynamic scan can reduce test data volume and test application time.

We then collected data on the scan cell usage of test patterns that could benefit from dynamic scan chains. The data in Figure 6 shows that several scan cells were never used after simulation of the first few fully specified test patterns. Thus, if we can identify these scan

cells, we should be able to place them in a single dynamic-scan segment of balanced scan chains and bypass them for the test patterns after the first 200. This single-partition dynamic-scan configuration is an attractive option in this type of situation.

Figure 6. Unused scan cells after a random-fill of the test vectors. Identifying unused scan cells is beneficial because dynamic scan would permit subsequent tests to bypass them.

Having more segments available in a dynamic-scan implementation improves the reduction in test data volume and test application time. The final goal of our research is to prove that by adding a few multiplexers and creating some partitions, dynamic scan could provide significant gains.

We use a hypergraph-based algorithm that evaluates test pattern scan-cell usage to determine scan chain segments and partitions.[7,8] The basis for this algorithm is an undirected graph having all scan cells as its nodes. Nodes in the graph are connected to each other if they contain specified values as part of a single test pattern. We complete the graph for all test patterns. Thus, if two scan cells have specified values in *n* tests, *n* edges between the two nodes will represent the scan cells. We then partition the resulting undirected graph to find "waists" (cut points with the fewest edges).

Figure 7 shows the reduction in test cost (data volume and application time) for the evaluated designs. We scaled the benefits of dynamic scan to the test set. The first few patterns used the entire scan chain and did not benefit from the dynamic scan configurations.

Figure 7. Data volume reduction. Dynamic-scan technology reduced test costs (data volume and test application time).

Figure 7 shows the results of three possible dynamic scan configurations using designs with four to 10 partitions. The results for a two-way split show the benefits of creating a segment that includes only the unused scan cells for the patterns that will benefit from dynamic scan. There are just two segments—one for the used scan cells and another for the unused cells.

The second set of results shows the maximum experimental reduction achieved by using a hypergraph-based algorithm to create more partitions. The maximum theoretical reduction achieved for the experiment reflects the fact that the scan can bypass every don't-care in the test patterns profiled in Figure 5.

//Author: Rather than summarizing points made in the article, please provide a conclusion that discusses ongoing work, future research directions, or implications of this work on other disciplines or areas.//

Conclusions

The results in this paper showed that simple modifications to existing scan technology can result in benefits in test application time and test data volume. The work presented focused on proving the technology. Further work is being performed in refining the dynamic scan technology. Efficient and better segment identification algorithms are being explored for the dynamic scan architecture.

Acknowledgments

We thank the many reviewers who provided constructive criticism that improved this article.

References

1. V.D. Agrawal, S.K. Jain, and D. Singer, "A CAD System for Design for Testability," *VLSI Design*, **Month** 1984, pp. 46-54.
2. S.P. Morley and R.A. Marlett, "Selectable Length Partial Scan: A Method to Reduce Vector Length," *Proc. Int'l Test Conf.*, IEEE CS Press, Los Alamitos, Calif., 1991, pp. 385-392.
3. S. Narayanan and M. Breuer, "Reconfiguration Techniques for a Single Scan Chain," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, no. 6, pp. 750-765.
4. S. Narayanan, R. Gupta, and M.A. Breuer, "Optimal Configuring of Multiple Scan Chains," *IEEE Trans. Computers*, vol. 42, no. 9, pp. 1121-1131.
5. I. Hamzaoglu and J.H. Patel, "Reducing Test Application Time for Full Scan Embedded Cores," *Proc. Int'l Symp. Fault Tolerant Computing*, IEEE CS Press, Los Alamitos, Calif., 1999, pp. 260-267.
7. G. Karypis and V. Kumar, *hMetis 1.5: A Hypergraph Partitioning Package,* tech. report, Dept. of Computer Science, Univ. of Minn., 1998; http://www.cs.umn.edu/~metis.
8. G. Karypis and V. Kumar, *Multilevel k-way Hypergraph Partitioning,* tech. report, Dept. of Computer Science, Univ. of Minn., 1998; http://www.cs.umn.edu/~metis.

Samitha Samaranayake *is a M.Eng student at the Massachusetts Institute of Technology. His research interests are in the area of Test Automation and Graph theory. Samaranayake received a S.B. in Computer Science from MIT in 2002. He is a member of Tau Beta Pi and Eta Kappa Nu. Contact him at samitha@mit.edu.*

Nodari Sitchinava *is a M.Eng. student at Massachusetts Institute of Technology. His research interests are in the area of theoretical computer science, in particular, graph theory, computational geometry and analysis of algorithms. Sitchinava received his S.B from Massachusetts Institute of Technology. He is a student member of MIT Chapter of IEEE. Contact him at nodari@mit.edu.*

Rohit Kapur *is a principal engineer at Synopsys. His research interests are in VLSI test. He received a PhD in computer engineering from the University of Texas at Austin. Kapur is a senior member of IEEE and a member of the IEEE Computer Society. Contact him at rkapur@synopsys.com.*

Minesh B. Amin *is a Staff Engineer at Synopsys. His research interests are parallel and distributed algorithms, verification and test. He received a Ph.D. from University of Minnesota -- Twin Cities. Contact him at mamin@synopsys.com.*

Thomas W. Williams *is a chief scientist at Synopsys. His research interests are in VLSI test. He received a PhD in electrical engineering from Colorado State University. He is a Fellow of the IEEE and a member of the IEEE Computer Society and the ACM. Contact him at tww@synopsys.com.*