Testing for Path Delay Faults Using Test Points *

S. Tragoudas N. Denny
Electrical and Computer Engineering Department
The University of Arizona
Tucson, AZ 85721

Abstract Path delay fault testing is often difficult due to the large number of paths that must be tested. Inserting controllable/observable points in the test architecture has been shown to be a viable method for reducing the number of paths that need to be tested in a circuit. Test points allow the tester to test subpaths of the circuit and then draw conclusions of the operability of the circuit based upon the delays of subpaths. We illustrate some of the limitations of current subpath testing procedures and illustrate some of the difficulties associated with unstructured test point placement. We give an implementation of test points embedded in a schan chain and present a new testing technique that is more accurate than the previous method. We also present a novel test point insertion approach that has reasonable test times and minimal impact on the hardware size and the operational clock.

1: Introduction

Path delay faults (PDFs) are circuit faults caused by some defect that results in some I/O path in the circuit having more or less delay than than specified in the design of the circuit. We are mostly concerned in the faults that produce delays that are longer than the expected delay. Unlike stuck-at faults which are static (permanent) faults, PDFs are faults only in the context of their operating environment. Specifically, a PDF exists only within a certain range of operational clock periods.

For a given path, there are two possible faults, a $0 \rightarrow 1$ fault and a $1 \rightarrow 0$ fault. Here the first value is the steady state value of the primary input of the path of interest, and the second value is the change on the primary input.

Path delay fault testing of the circuit is done in an enumerative fashion. Each potential fault is tested in isolation by using a set-up vector to create the preconditions for the transition, and a trigger vector to initiate the transition. To isolate the fault, the circuit must be stable after the application of the set-up vector and before the application of the trigger vector. A slow clock of period greater than the operational clock is used to time the application of the set-up vector. After applying the trigger vector, we use the operational clock to test for a delay in a transition along an I/O path. Delay faults are identified by observing the target output one operational clock period after the application of the trigger vector. If the target output has not changed from its value after the set-up vector, then the circuit is faulty.

The difficulty in enumerative testing for PDFs is two-fold. First, the test pattern generation (TPG) tool must accomplish the non-trivial task of generating a pair of test vectors

*This work was partially supported by NSF grant CCR-9815229.

that excite the desired path. Secondly, in most circuits of interest the number of I/O paths is prohibitively large.

Much research has been devoted to both difficulties. Publications [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] have targeted the first problem. Efforts in [14] [15] have addressed the difficulties of testing circuits with many paths.

One method of addressing both the number of paths and the testability of paths in a circuit, is the use of test points. Test points divide full I/O paths into subpaths. This has a double benefit in that there are fewer subpaths than full I/O paths, and it is generally much easier for the TPG tool to generate test vectors for subpaths than full I/O paths. The latter benefit is derived from using (controllable) test points to apply test vectors topologically near to the subpath of interest, thus requiring fewer line justifications to be performed by the TPG tool. Most testing benefits are the result of both controllable and observable properties. We will assume that all test points possess both such properties.

The remainder of this paper will focus on the limitations of testing with subpaths and identify two reasons for caution in placing test points. In Section 2 we review the subpath testing method of [15] and critique their test point placement algorithm. We discuss over-simplifications that lead to inaccuracies and objectives that the test point testing tool failed to address. In Section 3, we present our method for testing using primary inputs, primary outputs, and test points and illustrate the need for a careful placement of test points. In Section 4 we present an ILP formulation for placing test points so that the testing process is more efficient. In Section 5, we present experimental results on the ISCAS85 benchmark set that show the advantages of the proposed method. Finally, we conclude in Section 6.

2: PDF testing using test points

This section reviews the test point approach proposed in [15], outlines its limitations and analyzes hardware overhead related aspects. We first analyze the procedures proposed in [15] for testing with test points. After inserting test points, I/O paths that do not contain test points are tested in the same manner previously described in the introduction. An I/O path that contains at least one test point is divided into subpaths. Each subpath is delimited by a primary input and a test point and a test point and a primary output. In the case of more than one test point on a path, a subpath may also be delimited by a pair of test points.

Testing proceeds at the level of subpaths. For each subpath, the TPG tool produces the setup and trigger vectors, the point of observation, and the test clock. When developing the test vectors, the TPG tool is free to use some combination of primary inputs and (controllable) test points. The point of observation may be any of the primary outputs, or a (observable) test point. Any remaining test points and primary inputs are don't cares.

In testing a complete I/O path, the testing process requires a slow clock to be used for all set-up vectors and the operational clock to be used for all trigger vectors. In subpath testing, the slow clock is used for all set-up vectors, the operational clock for trigger vectors that test complete I/O paths, and additional test clocks for testing the subpaths.

The additional clocks are computed from a graph model of the circuit. The set of controllable points includes all test points and all primary inputs. Similarly, the set of all observable points includes all primary outputs and all test points.

Let a region of the circuit be defined as as all subpaths from some controllable point to some observable point. Using the traditional method in [15] each region can be timed by its own clock. Selecting one controllable point, one observable point, and accounting only for forward edges yields an approximate upper bound on the number of clocks as

$$test\ clocks = \frac{(N + inputs) \cdot (N + outputs - 1)}{2}$$

where N is the number of test points, *inputs* are the number of primary inputs and *outputs* are the number of primary outputs.

[15] assume unit delays on gates of the circuit and they observed that under this simplification many of the clocks will be computed to be the same value. For example if $\{1,2,2,1,5,5,5,3\}$ is the computed set of clocks, the actual number of clocks needed is only 4, $\{1,2,3,5\}$. Unfortunately, the unit delay model is not realistic and in practice the number of needed clocks is much higher.

Furthermore, an essential component of the testing method of [15] is the assumption that any subpath $X \rightarrow Y$ that has delay greater than that of the longest path from $X \rightarrow Y$ is faulty and implies that the circuit is faulty. While this assumption may be valid in eliminating some types of faults in the manufacturing process, it is inadequate for field testing. When testing equipment that has already been in operation, the goal is to determine whether or not a given component is faulty with respect to its operating environment. Whether an isolated subpath of the circuit exhibits increased delay is insignifigant. The only relevant test measure for PDFs is the timing of complete I/O paths.

We conclude that memoryless testing of isolated subpaths is inaccurate. To increase the accuracy of the testing process we develop a testing process that is both reasonable in application time and accurate in determining path delay faults.

Practical (non-ideal) test points have an impact on the host circuit. They require additional hardware, interface pins, and propagation delay. In most cases, directly controllable and observable test points are not feasible to implement. For even modest numbers of directly controllable test points, the external pin count becomes very large. Test points also incorporate bypass logic. In testing, the bypass logic routes signals to the test point hardware. In normal operation, signals propagate through the bypass logic. The total propagation delay of the path is then increased by the propagation delay of the bypass logic. A placement algorithm that is free to place multiple test points on a path may place multiple test points on the longest sensitizable path and effectively increase the operational clock period of the circuit.

One possible solution to the problem of numerous external pins is to implement test points as elements of a scan chain. Test points embedded into a scan chain are constructed with two state devices similar to the double-strobe flip-flop of [16]. Each test point can then store an element from both the setup and trigger vectors. The increase in propagation delay is not easy to address by hardware alone. We propose instead to use a placement algorithm that can control the number of test points inserted into a single path. We have developed one such algorithm and discuss it in detail in Section 4.

3: The proposed test method

This section assumes that tests points have been inserted and the set of subpaths has been defined. A method for inserting test points is described in the next section. To address the problems with subpath testing, we propose in this section a new physical test procedure.

To more accurately test a circuit using test points we must recognize the dependencies between subpaths and to some degree the environment in which the circuit operates. For PDF testing, the property of the operating environment that is of interest is the operational clock period.

For a complete I/O path, P, of subpaths $P_1, P_2, ..., P_n$, each subpath P_j has timing CP_j. The delay of the entire path must not exceed the operational clock, CP.

$$CP \geq \sum_j CP_j$$

We can deduce the required timing for one subpath, P_i, if the remaining $P_n - 1$ subpaths are known. To find the remaining CP_j, we must empirically measure the minimum delay where P_j operates correctly. We can empirically measure the delay of a subpath to a desired accuracy (limited by test equipment resolution) by using a binary search on the clock.

We illustrate the test process with the simple combinational circuit represented in Figure 1. Test point T_1 has a fanin cone and a fanout cone. The fanout cone contains fewer subpaths than the fanin cone. To keep test application time to a minimum we choose to apply our tests to determine the timing the of the fanout cone and deduce the timing of the fanin cone.

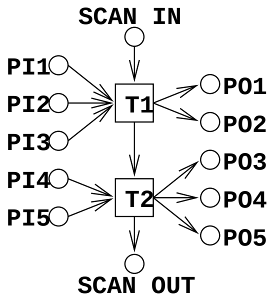

Figure 1. Test example.

We begin by querying the TPG tool for the set-up vector, the trigger vector, the points of controllability, and the point of observation for the subpath $T_1 \rightarrow PO_1$. In this simple example, there is only one point of controllability, T_1. We set the scan chain for controllability and shift in the trigger vector and then shift in the setup vector. We start the slow clock and wait as the circuit stabilizes. We tune our test clock to the operational clock period, CP. We apply the trigger vector and simulatneously start our the test clock. After the test clock, we observe PO_1 for a transition. If we see a transition, we test the subpath again. We retune the test clock for $\frac{CP}{2}$ and reapply the set-up vector. After the slow clock, we start the test clock and apply the trigger vector, observing PO_1 after the test clock. If this timing fails, we apply our test again with a test clock of $\frac{3 \cdot CP}{2}$. In this fashion we divide the test clock until we have reached the desired timing accuracy of subpath $T_1 \rightarrow PO_1$.

To fully test the region, we apply the same test technique to subpath $T_1 \rightarrow PO_2$. After determining all of the subpaths in the fanout cone of T_1 we use the maximum delay of all of the subpaths in the cone to deduce the timing of the subpaths in the fanin cone of T_1. Let CP_2 be the experimentally determined timing of region the fanout cone of T_1. Then $CP_1 \leq CP - CP_2$ for the circuit to be PDF free with the operational clock, CP, where CP_1 is the timing of the subpaths in the fanin cone of T_1.

For some complete path that is composed of subpaths in a set of regions $\{R_1, R_2, ..., R_n\}$, the worst case timing of the complete path is the sum of the worst case timing of each region, where the worst case timing of some region R_j is the maximum timing of any subpath in R_j. This observation leads to a simple optimization in the test schedule. When testing a region, we do not need to accurately determine the timing of every subpath in the region.

For example, consider a region of subpaths $P_1, P_2, ..., P_m$ in a circuit with an operational clock of 20ns. To determine the timing of a subpath within 0.1ns requires 8 steps in the binary search on the clock. To determine the worst case timing of the region, we start with subpath P_1 and use all 8 steps to determine the timing. We then begin testing P_2, but after only 6 steps (within 0.3ns accuracy) we observe that P_2 has a timing less than P_1. Since we are only concerned with the maximum timing of the region, we drop the remaining two steps in the binary search on the clock of P_2 and move on to P_3.

4: Algorithms for test point placement

We begin with the a general formulation for placing test points so that they form a single cut of the circuit digraph. This process is driven by the necessity to control the maximum number of test points on any circuit path. We formulate an integer linear program (ILP) for placing test points that meet this goal. The ILP will place test points so that no two test points are on the same path and such that the test application time is minimized under this constraint. After the test points are inserted two subcircuits are formed, the fanin subcircuit and the fanout subcircuit. If the test application time is still prohibitive, the ILP will be called recursively for the two subcircuits.

Our inspiration for the test point placement within one phase is from the operation of retiming for weighted state minimization as introduced in [17]. In the general case, ILP is a difficult problem. However, the formulation that we develop here is a special case of ILP that can be solved using a minimum cost flow network. (The details of this reduction of our ILP to a minimum cost flow network are omitted here due to space constraints.) Several polynomial time algorithms are known for solving minimum cost flow networks.

In the following, we use the standard graph model of a circuit. We model the circuit as a digraph, $G = (V, E)$. Gates in the circuit are nodes in V, and the interconnections between gates are the edges in E.

We begin by defining a cut as a placement of test points such that every I/O path contains exactly one test point. For a purely combinational circuit, we use as the basic case of a cut the placement of one test point on each output edge. (Every I/O path then contains one and only one test point.) We define an operation for moving test points such that applying this operation to a cut always yields another cut. For a given node v, if each out edge of the node has a test point, we may remove the test points on the out edges and place them on the in edges. For every path that contains v, and contains exactly one test point, we do not increase the number of test points contained in the path by such a move.

Next, we define a mapping of each node to a binary integer value $S(v) \in \{0, 1\}$, $v \in V$. We define $S(v)$ as the "shift" of the node $v \in V$. The value $S(v)$ maps to 1 if the node v has participated in a moving of test points from its out edges to its in edges, and maps to 0 otherwise. For our initial condition, all $S(v) = 0$. Observe that since all shifts occur from out edges to in edges then for all legal sequences of shifts, $S(v) = 0$ if v is an output node. We make an additional constraint that no sequence of legal shifts can shift a test point before an input node. Therefore, for any legal sequence of shifts, $S(v) = 0$ if v is an input node.

In performing a shift, we are subtracting a test point from the out edges and adding a test point to the in edges of a node. The number of test points on edge (u, v) after a shift is

$$t_{after} = t_{before} + S(v) - S(u)$$

where t_{before} is the number of test points on the edge (u,v) before the shift.

For the placement to be physically realizable, we must be careful not to make shift assignments such that t_{after} is negative. For each edge then, we have

$$S(u) - S(v) \leq t_{before}$$

We now have a set of constraints that defines the boundaries of the space of legal shifts.

We seek a cut with the property that it minimizes test application time. By placing a test point, tp, we break the circuit into a fanin and fanout cone of tp. We compute the number of paths from primary inputs to the testpoint, $fanin(tp)$, and the number of paths from the testpoint to the primary outputs, $fanout(tp)$. To minimize the test application time, we experimentally determine the timing of the cone that has the fewer paths using $\log(CP)$ tests per path and deduce the timing of the other cone. Our test application time for a single test point is determined by

$$\log(CP) \cdot fanin(tp) + fanout(tp)$$

if $fanin(tp) \leq fanout(tp)$ or

$$log(CP) \cdot fanout(tp) + fanin(tp)$$

otherwise.

In placing a cut we guarentee that all test points are independent of each other. The test application time of the entire circuit is a sum of the test application time required for each test point. Our ILP objective function is then

$$\min \sum_v S(v)((in\_edge\_test\_time(v)) - (out\_edge\_test\_time(v)))$$

where $in\_edge\_test\_time(v)$ and $out\_edge\_test\_time(v)$ are the sums of the test times for placing test points on all in edges and all out edges of node v, respectively.

After the ILP is solved, the cut may contain more test points than may be desirable. Indeed, some test points may not be effective at all. We remove unneeded test points by first computing the relative value of each test point. The relative value of a test point is the number of paths that the test point divides. We use the same path counting method introduced previously to compute $fanin(tp)$ and $fanout(tp)$ and assign for each test point, tp, the relative value

$$rv(tp) = fanin(tp) \cdot fanout(tp) - fanin(tp) + fanout(tp)$$

To remove unneeded test points, we remove all test points where $rv(tp) \leq 0$. If after this operation, the number of test points is still excessive, test points are removed in order of least relative value first until the number of test points is acceptable.

The ILP formulation for placing single test points in a single cut capitalizes on the independent property of each test point to place them in an optimal configuration. When placing multiple cuts, we no longer have test points that are independent of each other.

As a heuristic extension, we sacrifice global optimality and apply our ILP formulation recursively on the circuit. Each recursive call divides the circuit and increases the number of test points on an I/O path. For N recursive calls, some I/O path may have as many as $2^N - 1$ test points. Even for large circuits, many iterations are not likely to be needed (see Section 5).

5: Experimental Results

We ran our test point placement algorithm on the standard ISCAS85 benchmark set. Circuits c432 and c499 include implementation dependent XOR gates, and were excluded from our results.

For each circuit we executed our test point placement algorithm which we will call TPP. After inserting the test points with a single call of the ILP, we removed test points that did not reduce the test application time. In all cases except c6288, the number of test points was limited to 100 and a single cut. For the circuits where the ILP produced a cut of more than 100 test points, we used the simple pruning technique given at the end of Section 4. For comparison, we then executed the (Pomeranz-Reddy) PR placement algorithm to insert the same quantity of test points.

Table 1. Processing results and impact on host circuit

		PR		TPP	
Circuit	Test Points	Max	Longest	Most	CPU time (sec)
c880	11	5	5	1	2.08
c1355	8	2	2	1	3.20
c1908	30	5	5	1	5.64
c2670	26	5	5	1	9.07
c3540	58	9	6	1	12.93
c5315	99	7	7	1	26.86
c6288	431	42	41	7	103.39
c7552	100	5	5	1	55.92

Table 1 displays the impact of the test points on the host circuit. For the PR method, we detail both the maximum number of test points found on any I/O path in column 3 and the maximum number of test points found on a longest I/O path in column 4. The latter attribute has the effect of increasing the operational clock and reducing the performance of the host circuit. The maximum number of test points placed on any I/O path by our method is reported in column 5. The CPU times reported in column 6 are the solution times of the ILP using the standard LP_SOLVE linear program solver executed on a Sun ULTRA 5 workstation. [1]

Table 2. Comparison of test applications, scan chain shifts, and subpaths

	PR			TPP		
Circuit	Tests	Shifts	Paths	Tests	Shifts	Paths
c880	4,210	51,128	985	4,864	11,088	1,088
c1355	561,792	8,729,952	277,956	48,558	64,512	18,672
c1908	6,376	203,940	1,795	34,810	628,800	9,705
c2670	7,722	121,056	3,511	36,906	40,768	16,710
c3540	34,282	2,112,360	5,493	211,222	9,177,920	63,632
c5315	10,120	1,714,284	4,766	53,164	1,629,936	19,806
c6288	9,032	6,640,488	7,807	89,891	52,204,212	84,107
c7552	43,878	4,837,600	17,214	98,972	1,140,800	33,400

[1]The LP_SOLVE package is available via FTP at ftp.es.ele.tue.nl in directory /pub/lp_solve

The data in Table 2 displays the worst case test times and shift operations for both placement alogorithms assuming that all test points were embedded in a single scan chain and testing was performed using the method that we detailed in Section 3. In the case that our ILP inserts only a single cut, the objective function of the ILP simultaneously computes an optimal test schedule for the cut. However, in the case that some I/O path contains more than one test point, the dependencies between the test points present difficulties in computing optimal schedules. For computing these test schedules we used a greedy algorithm for determining the deduction regions.

The number of tests reported in columns 2 and 5 is the number of subpath tests needed to fully test the circuit for both $0 \rightarrow 1$ and $1 \rightarrow 0$ transitions. In our experiments, we used an operational clock of 20ns and an accuracy of less than 0.1ns. Each subpath that must be determined (not deduced) required 8 test applications. The worst case occurs only when all subpaths outside of the dedcution regions require all 8 test applications to fully determine their timing. Such a case is not likely in practice, but provides an upperbound for comparison.

The shift data presented in columns 3 and 6 is the worst case number of shift operations required to load the scan chain throughout the testing process. In computing the number of shift operations, there are three possible cases. **Case 1**: The point of observation is a primary output or test point and all controllable points are primary inputs. **Case 2**: The point of observation is a primary output and the set of controllable points includes one or more test points. **Case 3**: The point of observation is a test point and the set of controllable test points includes one or more test points.

Case 1 is ideal in that no shift operations need be applied. The setup and trigger vectors are applied directly to the primary inputs. Case 2 requires the tester to shift both the setup and trigger vectors into the scan chain. Once the vectors are loaded, they may be used to test a subpath many times. Case 3 occurs only when there are dependencies between test points in the same scan chain. The difficulty introduced in this case is that the scan chain must first operate in controllable mode to apply the setup and trigger vectors, but then must be rapidly switched into observable mode and clocked to capture the result of the subpath test. For our two state test point, this has the side effect of overwriting the previously loaded setup vector. Thus before each test of a subpath, the tester must reload the setup vector.

Our test point placement algorithm does not explicitly target path reduction. We do however present the number of subpaths for both placement algorithms in columns 4 and 7. This data is presented for completeness only.

When both tests and shift operations are considered, our ILP insertion process performs comparable to (or better than) the PR insertion method for c880, c1355, c1908, c2670, c5315, and c7552. The results clearly show that our ILP based placement has a signifigantly less shift per test ratio than the PR method. Our ILP is able to achieve these test schedules with typically 1/5th of the performance impact of the PR method.

In the case of c6288, we report in Table 2. a serial test schedule for a single scan chain. Our ILP inserted 7 partial cuts into the circuit and in practice, these partial cuts create natural boundaries for parallelizing the test schedule. We estimate the parallelized test schedule to be equivalent to a serial test schedule of approximately 30,000 tests and 7,450,000 shifts.

6: Conclusions

We analyzed some limitations of an existing test point placement approach. We presented a new, accurate test procedure for PDFs. We also presented a novel test point placement algorithm that uses an ILP to both bound the number of test points on an I/O path and minimize test application time. We presented experimental results for comparing the performance of our ILP to that of the greedy placement algorithm suggested by [15]. From these results we concluded that our ILP insertion method performs comparable to [15] in most benchmarks with typically only 1/5th of the performance impact of the PR method.

References

[1] D. Bhattacharya, P. Agrawal and V.D. Agrawal, "Delay Fault Test Generation for Scan/Hold Circuits using Boolean Expressions," Proceedings 29th Design Automation Conference, pp. 159–164, 1992.

[2] S. Bose, P. Agrawal and V. D. Agrawal, "Generation of Compact Delay Tests by Multiple Path Activation", International Test Conference, pp. 714–723, 1993.

[3] Chin–Ang Chen, Sandeep K. Gupta, "Test generation for path delay faults based on satisfiability", 33^{rd} Design Automation Conference, 1996.

[4] K.-T. Cheng and H.-C. Chen, "Generation of High Quality Non–Robust Tests for Path Delay Faults", 31^{st} Design Automation Conference, 1994.

[5] R. Dechsler, "BiTeS: A BDD based Test Pattern Generator for Strong Robust Path Delay Faults," Proceedings of Euro-DAC'94, pp. 322–327, 1994.

[6] K. Fuchs, M. Pabst, and T. Roessel, "RESIST: A Recursive Test Pattern Generation Algorithm for Path Delay Faults," Proceedings of Euro-DAC'94, pp. 316–321, 1994.

[7] A. Krstic, K.-T. Cheng, and S.T. Chakradhar, "Identification and Test Generation for Primitive Faults," Proceedings of the International Test Conference, pp. 423–432, 1996.

[8] Y. K. Malaiya and R. Narayanaswamy, "Testing for timing faults in synchronous sequential integrated circuits", Proc. Int. Test Conf., pp. 560-571, Oct. 1983.

[9] I. Pomeranz, S. M. Reddy and P. Uppaluri, "NEST: A Nonenumerative Test Generation Method for Path Delay Faults in Combinational Circuits", IEEE Trans. on CAD, Vol. 14, No 12, pp. 1505-1515, Dec. 1995.

[10] S. M. Reddy, M. K. Reddy and V. D. Agrawal, "Robust Tests for Stuck-Open Faults in CMOS Combinational Logic Circuits", in Proc. Int. Symp. on Fault-Tolerant Computing, pp. 44-49, June 1984.

[11] M. H. Schultz, K. Fuchs and F. Fink, "Advanced Automatic Test Pattern Generation Techniques for Path Delay Faults", Proc. Int. Symp. on Fault-Tolerant Computing, pp. 45-51, June 1989.

[12] J. Saxena and D. K. Pradhan, " A Method to Derive Compact Test Sets for Path Delay Faults in Combinational Circuits", International Test Conference, pp. 724–733, 1993.

[13] S. Tragoudas, D. Karayiannis, "A Fast Nonenumerative Automatic Test Pattern Generator for Path Delay Faults", to appear in IEEE Transactions on CAD, July 1999.

[14] A. Krstic, K. Cheng, "Resynthesis of Combinational Circuits for Path Count Reduction and for Path Delay Fault", Journal of Electronic Testing: Theory and Applications vol 11, pp. 43–54, 1997

[15] I. Pomeranz, S. Reddy, "Design for Testability for Path Delay Faults in Large Combinational Circuits Using Test Points", IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, vol. 17, no. 4, pp.333–343, April 1998.

[16] B. Dervisoglu, G. Strong, "Design for Testability: Using Scanpath Techniques for Path-Delay Test and Measurement", International Test Conference, pp. 365–374, 1991

[17] C. Leiserson, J. Saxe, "Retiming Synchronous Circuitry", Algorithmica, vol 6, pp. 5–35, 1991