INTERNATIONAL JOURNAL OF RESEARCH
In Advanced Engineering Technologies

Volume 4, Issue 2 JULY 2015

Analysis of Two Pattern Test Cubes for Transition Path Delay Faults in Digital Circuits

[1]**N. MADHAVI**,, PG Scholar in VLSI Design,
[2]**CHAKRI SREEDHAR**, M.Tech, Asst.Professor, ECE Department,
[1] madhvinagapatla @gmail.com,
[2]chakri.sreedhar@gmail.com,

ABSTRACT:

A method is presented for identifying primitive path-delay faults in non-scan sequential circuits and generating robust tests for all robustly testable primitive faults. It uses the concept of sensitizing cubes introduced in an earlier paper and a new, more efficient algorithm for generating them. Sensitizing cubes of the next state and output logic are used to obtain static sensitizing vectors that can be applied to the non-scan sequential circuit as part of a vector-pair. These vector-pairs are also used in deriving robust tests. Initializing sequences from a reset state and sequences that propagate fault effects from flip-flops to primary outputs are also generated. The proposed procedure unspecified them gradually to obtain a series of test sets with increasing numbers of unspecified values and decreasing path lengths. Experimental results also indicate that filling the unspecified values randomly (as with some test data compression methods) recovers some or all of the path lengths associated with detected path delay faults. The procedure uses a matching of the sets of detected faults for the comparison of path lengths.

Keywords—Full-scan circuits, path delay faults, test cubes, transition faults, two-pattern tests.

I. INTRODUCTION

Delay testing is becoming increasingly important because of the need to operate digital circuits at the highest possible speeds. The circuit under test is assumed to have been tested for logical faults and therefore functionally correct. Faulty behavior is modeled by delay faults, the most widely used model being the path delay fault. The presence of a path delay fault increases the propagation delay along the faulty path beyond the limit for correct operation. A test for a path delay fault propagates a signal transition along the path and checks whether the final value reaches the destination at the appropriate time. Delay testing of combinational circuits has received considerable attention in the literature, Since delay faults are often the result of variations in the fabrication process, we must be able to detect individual faults independent of actual delay values in the rest of the circuit. Tests that satisfy this condition are called robust tests [2]. Several techniques have been developed for generating robust tests for delay faults in combinational circuits [2, 3, 4], but the fault coverage obtained is often quite low. A weaker class of tests, called validatable non-robust (VNR) tests, has been proposed to test faults that are not robustly testable [3, 5].A VNR test detects the target fault if a set of other faults is also tested for and shown to be absent. Even with the use of VNR tests, all faults in a circuit may not be testable.

Achieving complete testability of asynchronous circuits has long been recognized to be a difficult problem since these circuits must be hazard-free. Hazard-free synthesis methods frequently introduce redundant or non-prime product terms, resulting in circuits which are not fully testable. Thus, ensuring hazard-free behavior

and at the same time achieving complete testability seem to be contradictory requirements. However, our aim in this paper is to show that hazard-free completely testable asynchronous multi-level circuits can be easily synthesized, in some rare cases requiring some extra control inputs.

In order to ensure high reliability of a circuit, one must test both its logical and temporal behavior for correctness. Physical defects may increase the propagation delays along different paths, giving rise to delay faults. Delay faults can be categorized according to two models: gate delay faults and path delay faults. The gate delay fault model models excessive delay limited to just one gate, whereas the path delay fault model models excessive delays along a whole path from an input to an output. Therefore, the path delay fault model is more comprehensive; however, it may require more time for test generation because the number of paths is usually much larger than the number of gates.

Delay faults are generally tested by two-pattern tests. For path delay faults, these tests launch a transition at the input of the path to see if the desired transition reaches the output of the path within the specified time. A two-pattern test is called robust if arbitrary delays elsewhere in the circuit cannot invalidate it. A robust test can be further categorized into a hazard-free or non hazard-free test. For a hazard-free robust test, no hazards can occur on the tested path irrespective of the delay values elsewhere in the circuit. This is the most stringent fault model. Hazard-free robust path delay fault testability of a circuit also implies testability under other fault models, such a stuck open. Since it is known that robust testability of general circuits is usually quite low, many syntheses for testability methods for this fault model have been presented. However, these methods are not geared towards hazard-free implementations that are required for asynchronous circuits.

II. LITERATURE SURVEY

This section describes the background needed for describing the proposed procedure.

The test sets considered in this paper are broadside test sets that were generated by the test generation procedure described in. The procedure targets path delay faults that are associated with the longest paths, and produces strong non-robust tests.

Faults that must be tested to guarantee timing correctness of a circuit are called primitive faults. Such faults were defined in for combinational circuits in terms of sensitizability of paths. We shall use a functional definition instead, so that it applies to both combinational and sequential circuits.

Definition 1:An MPDF F is primitive if there exists a vector-pair for which the circuit with only the fault F present may produce an observable output different from that produced by the fault-free circuit, and there is no F 0 F with the same property.

The following lemma follows directly from

Lemma 1: An MPDFF in a combinational circuit is primitive if and only if it is non-robustly testable and no sub fault of F is non robustly testable.

Our method of identifying primitive faults uses the concept of sensitizing cubes. A cube is defined as a subset of input literals. A cube can be represented by the values assigned to the inputs, or as a product of literals. Thus, a cube corresponds to a set input vectors, each of which will be referred to as a vertex. The vertices in a cube are said to be covered by the cube. A vertex in a cube that is not covered by any other cube is called an essential vertex. Each primary output of the circuit will be treated separately.

Definition 2:A minimal set of input values necessary to produce a specific output value (often called mandatory assignments) is called a sensitizing cube of the output. There are two sets of sensitizing cubes associated with each output of a circuit: sensitizing 0-cubes and sensitizing 1-cubes.

Definition 3: A path π is associated with a sensitizing cube q if (1) it sets every side-input to the non-controlling value when the on-path input is non-controlling and (2) no side input has a controlling value when the on-path inputs are controlling. We shall refer to q as a sensitizing cube of π.

Sensitizing cubes can be determined by tracing back from the output and assigning signal values as follows: If a controlling value is necessary to produce the desired gate output, assign the controlling value to a gate input that has not been chosen before. If non controlling values are required, assign it to all gate inputs. Repeat the process until primary inputs are reached. Justify all assigned line values, making only necessary assignments, and determine all implications. A conflict-free assignment of input variables obtained in this manner corresponds to a sensitizing cube.

For each sensitizing cube, the associated paths are identified using using Definition 3 as follows: For gates with one or more inputs with the controlling value, all gate inputs with the controlling value are on-path inputs of the same multipath. For gates with only non controlling value inputs, only one of the inputs is selected for the path. All sensitizing cubes and the associated multipath can be found by a depth-first traversal of the circuit using the above method.

Example 1:Consider the circuit (from [14]) and the set of signal values shown in Fig. 1(a). This set of necessary signal values to make f =1 is obtained by selecting the first (upper) input of each gate for assigning the controlling value. The input assignment corresponds to the sensitizing 1-cube.

$q_1 = \bar{a}\bar{b} = 0\,0\,x$

associated with paths $a357f, b257f$ and $\{a1457f, b1457f\}$. Similarly, assigning 1 to the lower input of gate 7, we get

$q_2 = \bar{c} = x\ x\ 0$

which sensitizes path $c67f$ to 1. The sensitizing cubes and the paths associated with them, obtained by justifying $f = 0$ are:

$1\ x\ 1$: $a357f$ and $c67f$.

$x\ 1\ 1$: $b257f$ and $c67f$.

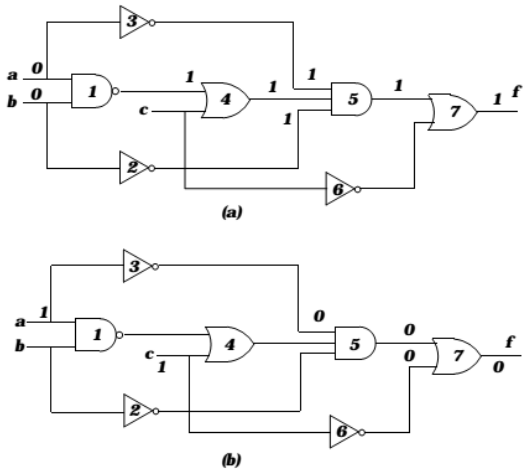

Figure 1. Finding sensitizing cubes.

Fig. 1(b) shows the value assignments that produce the sensitizing 0-cube 1x1. The cube x11is obtained by assigning 0 to the third input of gate 5. Note that assigning 0 to the middle input of gate 5 results in a conflict and does not produce a sensitizing cube. The example shows that a sensitizing cube may be associated with more than one multipath. A multipath may also have more than one sensitizing cube for the same output value.

The above method of computing sensitizing cubes eliminates the need for obtaining collapsed form expressions of the outputs. Although the worst case time complexity of the two methods may be the same, the memory requirements of the new method are expected to be lower. Unlike the earlier method, cubes that cannot be sensitizing cubes are not generated at all, resulting in some speed-up.

For the sake of convenience, we shall not explicitly specify the direction of transition for faults. A fault on a multipath π will refer to the MPDF on π for a rising or falling transition at the destination, depending on whether we are considering sensitizing 1-cubes or 0-cubes. We shall say that the path π is primitive when the fault on the path with the implied direction of transition is primitive.

Lemma 2: Every essential vertex of a sensitizing cube statically sensitizes a primitive fault.

Lemma 3: Let v be a common vertex of a set $Q = \{q_1, q_2, ..., q_k\}$ of sensitizing cubes, where $k > 1$. The multipath Π sensitized by v is primitive, if and only if (1) no $q_i \in Q$ has an essential vertex, (2) no proper subset of Q has a common vertex and (3) no proper subset of Π is static sensitizable.

Primitive faults in combinational circuits are identified by finding MPDF's that are statically sensitized by vertices satisfying Lemmas 2 and 3. Since a path may have a number of sensitizing cubes, some faults may be identified as primitive more than once. There may also be faults F_i, F_j that satisfy condition (1) of Lemma 3. If $F_i \subseteq F_j$, F_j must be deleted from the set of primitive It has been shown in [22] that all primitive faults in combinational circuits can be identified using Lemmas 2 and 3.

III.SEQUENTIAL TEST GENERATION

If the local test generation is successful the propagation part is called if the fault effect has reached a PPO. In that case the propagation of SEMILET is performed until a PO is reached by using forward time processing. The fault location is not needed to be known by SEMILET because a slow clock is assumed for the propagation phase and thus the fault does not occur in these time frames. During the propagation phase to a PO it is possible that some values at PPIs are not justified directly. For this justification task the propagation justification phase is called. Since this process is performed in backward direction, attaching all of the time frames used during propagation, finally the fast clock time frame is reached. Then the local test generation is called for performing the propagation justification task for the fast clock time frame resulting into an init state to be synchronized.

The synchronization is again performed by SEMILET. For the synchronization a slow clock is used and thus the faulty machine does not differ from the good machine. Note that the propagation justification and the synchronization phases both are performed by using reverse time processing while the propagation phase uses forward time processing.

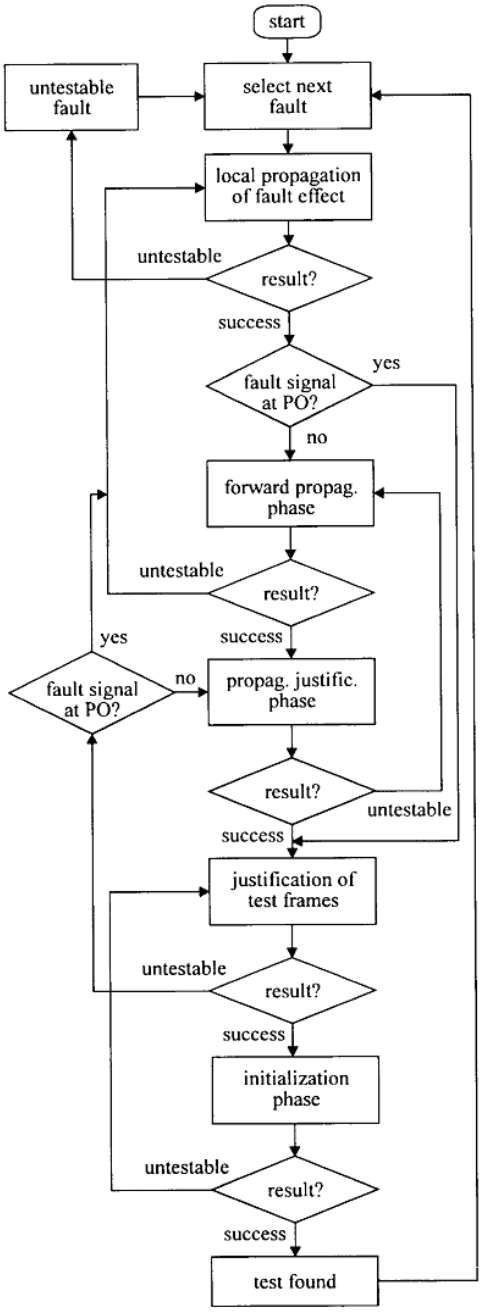

Fig 2.The extended FOGBUSTER algorithm

IV.EXPERIMENTAL RESULTS

The combination of TD gen and SEMILET has been used to generate robust tests for delay faults in a number of the ISCAS89 sequential benchmark circuits. Each line in each circuit was to be tested for a StR as well as a StF fault. As fault simulation is performed after each successful generation of a test, faults that were additionally tested by the generated patterns were not explicitly targeted by the test pattern generator. Table 3 shows the results of these experiments. For each circuit the total number of tested faults (second column), the number of un testable faults (third column) and the number of faults where test pattern generation was abandoned (fourth column) is shown. Test pattern generation was aborted after either 100 backtrack for the local test pattern generator, or 100 backtrack for the sequential test pattern generator. The fifth and sixth column show the total number of patterns generated and the total time needed for test pattern generation, on a Sun Sparc 10 Station, in seconds. The number of patterns generated as shown in the fifth column includes the patterns needed for initialization and propagation. It is remarkable that for some circuits the number of untestable faults is quite high. Although some of these faults are combinationally redundant, a large part of these faults is only sequentially untestable, i.e. a test for the delay fault in the combinational part can be generated, but the fault effect cannot be propagated to a PO in the propagation phase, or the pattern cannot be initialized. A lot of these sequential untestables are due to the robustness criterion, that does not allow to specify a value for a PPO that changes it's value in between the initial and final time frame of the local test pattern generation: because of the fast clocking, the stabilization of the final value cannot be guaranteed. Only the values of the PPOs that have an equal initial and final value, without a hazard, can be specified by TDgen to SEMILET. The final value of PPOs that show a transition or a hazard cannot be specified robustly (without any chance to test invalidation). These values are handed over by TDgen as a special kind of don't care, that is unjustifiable: SEMILET must assume a fixed, but unknown value is present. Nevertheless it has to propagate the fault effect without this knowledge. This is a hard task if many of these values are present, resulting in a high number of faults that is untestable by this logic.

Future work will include the extension of the robust local test pattern generation to an approach in which the arrival and stabilization times of all signals are calculated, allowing a more precise indication of signal values at certain times. This will make the task of propagation of the fault effect easier, thereby making robustly untestable faults testable.

V.CONCLUSIONS

This paper presents an approach to robust gate delay fault ATPG for circuits without a scan path. It relies on coupling TDgen, a combinational test generator for gate delay faults with SEMILET, a sequential test generator using the FOGBUSTER-algorithm. Experimental results on benchmark circuits show that the number of untestable faults due to a strong robust delay fault model is large. This number is expected to be significantly decreased by using a non-robust fault model.

REFERENCES

[1] G.L. Smith, “Model for delay faults based upon paths”,Proc.Int. Test Conf.,pp. 342-349, Nov. 1985.
[2] C.J. Lin and S.M. Reddy, “On delay fault testing in logiccircuits,”IEEE Trans. on CAD, pp. 694-703, Sept. 1987.

[3] C. Lin, S.M. Reddy and S. Patil, "An automatic test pattern generator for the detection of path delay faults,",Proc. Int.Conf. on Computer Aided Design, pp. 284-287, Nov. 1987.
[4] M.H. Schultz, K. Fuchs and F. Fink, "Advanced AutomaticTest pattern Generation techniques for path delay faults,"
Proc. Int. Symp. on Fault-Tolerant Computing," pp. 445-51,June 1989.
[5] S. Devadas and K. Keutzer, "Validatable non-robust delayfault-testable circuits via logic synthesis," IEEE Trans. onCAD, vol. 11, pp. 1559-1573, Dec. 1992.
[6] K.-T. Cheng and H.C. Chen, "Delay testing for non-robust untestable circuits,"Proc. Int. Test Conf., pp. 954-971, Oct.1993.
[7] I. Pomeranz, S.M. Reddy and P. Uppalpuri, "NEST: A nonenumerative test generation method for path delay faults in combinational circuits,"IEEE Trans. on CAD, vol. 14, pp.1505-1515, Dec. 1995.
[8] W.K. Lam, A. Saldanha, R.K. Brayton
and A.L. Sangiovanni-Vincentelli, "Delay fault coverage and performance tradeoffs,"Proc. Design Automation Conf., pp.446-452, June 1993.
[9] W. Ke and P.R. Menon, "Delay-Verifiability of Combinational Circuits based on Primitive Faults," Proc. Int. Conf.on Computer Design, pp. 86-90, Oct. 1994.
[10] W. Ke and P.R. Menon, "Synthesis of delay-verifiable combinational circuits",IEEE Trans. on Computers., vol. 44, pp.213-222, Feb. 1995.
[11] D.B. Armstrong, "On finding a nearly minimal set of fault detection tests for combinational logic nets,"IEEE Trans. Electron. Computers, vol. EC-15, pp. 66-73, Feb. 1966.
[12] M. Sivaraman and A. J. Strojwas, "Primitive path delay fault identification,"Proc. Int. Conf. on VLSI Design, pp. 95-100, Jan. 1996.
[13] A. Krstic, K.-T. Cheng and S.T. Chakradhar, "Identification and test generation for primitive faults,"Proc. Int. Test Conf.,pp. 423-432, Nov. 1996.
[14] R. Tekumalla and P.R. Menon, "Test generation for primitive path delay faults in combinational circuits," pp. 636-641,Proc. Int. Conf. on Computer Aided Design, Nov. 1997