

Subscripted D-Algorithm

Alok Doshi

Abstract-- The main idea behind subscripted D-Algorithm is to reduce the test vector set and to increase the speed of test generation. To achieve this, the algorithm uses the technique of sensitizing multiple paths at the same time. It constructs a single test pattern to simultaneously test as many faults as possible. The time required to generate this test pattern is normally the same as the time that the D-Algorithm takes to find a single test. So the overall performance of the subscripted D-Algorithm is found to be much better compared to Roth's D-Algorithm.

I. INTRODUCTION

The subscripted D-Algorithm is also referred to as the AALG[1]. The AALG was introduced to accelerate the test generation process by trying to completely control the submodule under test. Flexible control signals are assigned to all the inputs of the submodule and are propagated backward toward the primary inputs of the circuit, while a flexible observation signal is propagated from the submodule's output to the primary outputs of the circuit. The multiplicity of the flexible signals causes conflicts. These conflicts are handled differently from the conflicts caused by fixed signals. Most of the conflicts are solved by reducing the flexible signal to a fixed value signal.

This algorithm generally yields a set of tests which detect a specific subset of faults located in a given submodule. Test patterns for multiple faults can be generated with the same computational effort as for single faults. The computational burden of ATPG increases with the amount of fan-out present in the circuit. The same is true for AALG but, the algorithm

becomes more effective if the number of fan-ins present in the circuit increases.

II. THE SUBSCRIPTED D-ALGORITHM

A submodule is defined as a group of one or more gates of the type AND, OR, NOR, NAND, and NOT. The input-output behavior of the submodule should be known. But if the submodule consists of only a single gate, the behavior of the gate can immediately be considered in order to activate and observe a fault located at one of the terminal lines of the submodule. For simplicity, submodules will be considered to be single gates to explain the algorithm. A signal D_j is assigned to each line connected to the gate under test. For an n -input gate, $D_1, D_2, \dots, D_n, D_0$ are assigned respectively to the n inputs and the output of this gate. D_j ($j=1, \dots, n$) is a flexible signal which can represent either 0 or 1 independently of the other input assignments regardless of whether the circuit is faulty or not. Likewise, D_0 can represent either 0 or 1 depending on the functional behavior of the gate under test. The three steps constituting the algorithm are [1];

1. D_0 -propagation to the circuit outputs (observation paths).
2. D_j -propagation to the circuit primary inputs (control paths).
3. Line justification.

A. D_0 Propagation

This is the first step of the algorithm in which a path is

sensitized from the gate output to the primary output. This procedure is similar to the D-propagation in Roth's algorithm. The only difference is that D_0 does not represent a signal that identifies the failure state of the circuit (In case of DALG a 'D' and a 'D-bar' represent the two different states of the circuit). If D_0 is fixed to the logical value 0 by setting the inputs to appropriate values, and then to 1 with another set of input values, both faults stuck-at-1 and stuck-at-0 at the output of the gate under test will be detected.

B. D_j Propagation

This is the second step of the algorithm in which a maximum number of D_j 's are propagated from inputs of submodule to the primary inputs so as to establish as many control paths as possible. Each D_j can either represent a 1 or a 0. In order to prevent the search for such paths from being exhaustive, all potential paths of interest are examined simultaneously by allowing every flexible signal D_j to replicate after crossing a gate. The basic gates, AND and OR share the same property that their output will be D_j if all of their inputs can be connected to a D_j chain. If some of the inputs are driven by appropriate enable signals (1 in the case of AND) the output will still be D_j if the remaining inputs contain D_j 's. Any D_j becomes D_jN (complement of D_j) after crossing through either a NAND, NOR or NOT gate. Figure 1[1] shows how the signal D_3 propagates through gate 15. D_3 proliferates and becomes D_3N (complement of D_3) when reaching gates 9 and 10. The process is repeated until the primary inputs are reached. The proliferation of D_j 's increases the likelihood that some replicas reach primary inputs while the remaining ones which are propagated on sterile paths are fixed to 0 or 1.

The main reason for conflicts is the reconvergence of two different flexible signals: a gate is assigned a D_j -signal while some of its inputs are assigned incompatible signals D_i ($i \neq j$). This leads to a decision as to which flexible signal is to control the reconvergence gate. The unwanted replica is fixed

to 0 or 1 and the effect of this change is felt throughout the circuit up to a certain depth. Therefore, to resolve a conflict, at least one branch of the tree constituted by the paths propagating replicas is deleted. In the most unfortunate cases, a whole path is lost, thus allowing only a restricted control over the corresponding input of the gate under test.

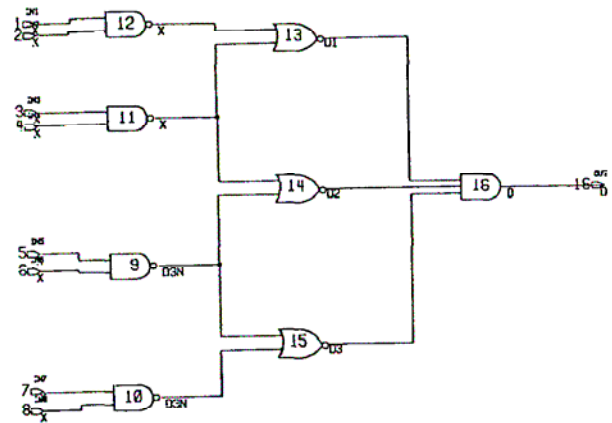


Figure 1. D_3 -propagation

C. Line Justification

This is the last step of the algorithm which analyses each gate that is assigned a 0 or 1 in order to insure the functional behavior of the gate. Input-output consistency involves a careful study of all consequences of a line assignment and multiple solutions must be examined to insure a minimum desensitization of the paths already established. A replica is forced to a fixed value 0 or 1 only when there is no other alternative. This is because the yield in tests over DALG is halved for each D_j lost by annihilation.

III. CONFLICTS ENCOUNTERED IN AALG

The performance of AALG is mostly determined by conflict handling in the area surrounding the gate under test. In generating a test set, a situation may arise that interferes with the path sensitization process; when this happens, paths can be incompletely established and do not reach the circuit's primary inputs as at the D_j propagation stage. Sensitized paths cannot propagate further until the conflict is resolved. The same situation arises in the line justification step due to which there may be desensitization of paths which are completely established. In any case, a sensitized path may abort. Figure 1[1] illustrates a conflict " D_3N vs. D_2 ". This type of conflict is encountered exclusively during the D_j -propagation phase. Here gate 16 is being tested and none of the paths is entirely established. In an attempt to push backwards the D_2 -frontier, gate 14 detects that one of its inputs (gate 9) is already assigned D_3N . It is therefore impossible to control gate 14 independently of the other inputs (13 and 15). An arbitrary decision must be made which will desensitize a portion or all of an existing path. The next step (figure 2[1]) shows a decision which unfortunately destroys all potential D_2 -paths. Gate 14 is assigned the logical value 1, thus insuring that any change at one of the inputs 13, 14 or 15 is transmitted to the output of gate 16. Now the D_j -propagation step can resume and remaining flexible signals D_1 and D_3 are propagated to the circuit's primary inputs. Another type of conflict is encountered in the line justification step. It is illustrated in Figure 3[1]. An attempt to justify gate 14 results in assigning the logical value 0 to both gates 9 and 11. These gates cannot propagate flexible signals anymore. Fortunately, a logical value 0 at both gates 9 and 11 does not interfere with the propagation of D_1 through gate 13 and D_3 through gate 15. The fragments of paths (12-13) and [10- 15) remain sensitized. The example shows that it is sometimes impossible for some sensitized paths to coexist.

IV. FAULT DETECTION USING AALG

Fault detection is explained with the help of circuit of Figure 4[1]. Gate 47 is tested using AALG. Four control paths are established allowing 16 different input combinations to be applied directly to gate 47. The set of tests at the primary inputs is;

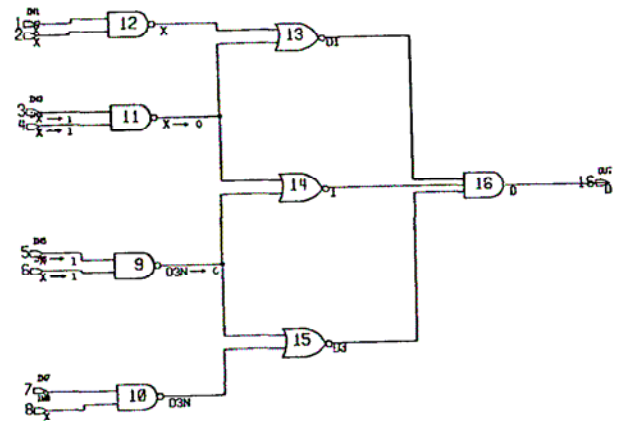


Figure 2. Resensitization of paths (13-16) and (15-16).

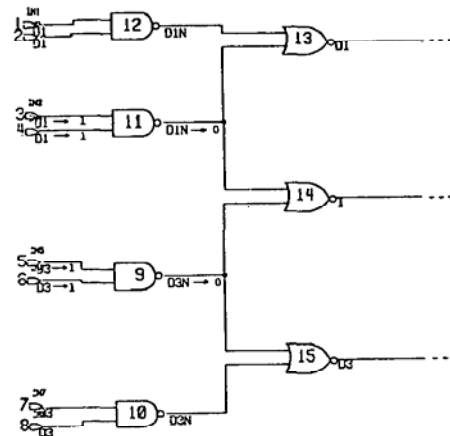


Figure 3. Snapshot of the line justification phase.

(0 D_1 0 D_2 0 D_3 0 0 D_4N D_4N 0 0 0)

Now we examine the single faults detected by the cluster of test patterns. A stuck-at-0 fault located at the output of gate 47 requires all inputs of gate 47 (40, 41, 42, 46) to be forced to the logical value 1 which in turn implies $D_1=D_2=D_3=D_4=1$. Hence a test for gate 47 stuck-at-0 can be derived and is explicitly indicated by the primary input assignments;

$$(0\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0)$$

A similar reasoning concludes that (0 0 0 - 0 - 0 0 - - 0 0 0) is one of the test patterns for gate 47 stuck-at-1. The set (0 D_1 0 D_2 0 D_3 0 0 D_4N D_4N 0 0 0) represents also a test for the following faults;

- gate 40 s-a-0 |
- gate 41 s-a-0 | _ (0 1 0 1 0 1 0 0 0 0 0 0 0)
- gate 42 s-a-0 | /
- gate 46 s-a-0 |

- gate 40 s-a-1 _ (0 0 0 1 0 1 0 0 0 0 0 0 0)
- gate 41 s-a-1 _ (0 1 0 0 0 1 0 0 0 0 0 0 0)
- gate 42 s-a-1 _ (0 1 0 1 0 0 0 0 0 0 0 0 0)
- gate 46 s-a-1 _ (0 1 0 1 0 1 0 0 1 1 0 0 0)

Hence all the faults on the inputs and outputs of the gate are detected with one test pattern.

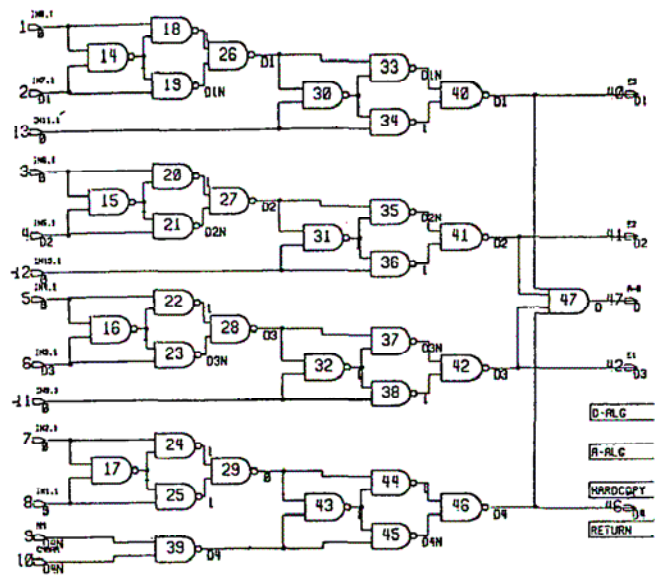


Figure 4. Testing of a gate using AALG (here gate 47 is tested).

V. IMPLEMENTATION

The AALG is implemented in 3 phases. In the first phase, different independent D_j signals are propagated from primary outputs to primary inputs. Multiple flexible paths can cross the whole circuit. Most faults along any of these paths can be activated and observed. This gives high fault coverage as these paths span a large region of the circuit. Two test patterns a '0' on all D_j 's and '1' on all D_j 's cover all faults on those paths. In the second phase, specific gates are tested. After the second phase, the remaining faults are scattered throughout the circuit. A technique called 'gang testing' is implemented for simultaneously detecting most of those faults. In this technique, each remaining fault is made observable at a different primary output till there are no more outputs. These faults are then sensitized and detected.

VI. CONCLUSION

The subscripted D-algorithm reduces the size of the test vector set and increases the speed of test generation as it generates a single test pattern for many faults. It is possible to

achieve fault coverage of almost 50% with just two test patterns during phase one of implementation. Sometimes there are conflicts which have to be resolved because of which the coverage might drop.

VII. REFERENCES

[1]. C. Benmehrez and J. F. McDonald "Measured Performance of a Programmed Implementation of the Subscripted D-Algorithm" Proc. of the 20th Design Automation Conference pp. 308-315 1983.

[2]. C. Benmehrez and J. F. McDonald "Test Set Reduction Using the Subscripted D- Algorithm" Proc. of the International Test Conference pp. 115-121 1983.

[3]. M. Ladjadj and J. F. McDonald "Benchmark runs of the Subscripted D-Algorithm with Observation path mergers on the Brglez-Fujiwara Circuits" Proc. of the 24th Design Automation Conference pp. 509-515 1987.

[4]. Michael Bushnell and Vishwani Agrawal "Essentials of Electronic Testing for digital, memory and mixed-signal VLSI circuits" Kluwer Academic Publications 2001.