The Pitfalls of Necessary Assignments

Wu-Tung Cheng, Rob Thompson, Aiman El-Maleh, Don Ross and Janusz Rajski
Mentor Graphics Corporation
8005 SW Boeckman Road
Wilsonville, Oregon 97070

contact: wu-tung cheng, tel: 503-6851078, fax: 503-6851654
email: wu-tung_cheng@mentorg.com

1.0 Introduction

It has been shown that, finding necessary assignments during a search process, such as Automatic Test Pattern Generation (ATPG) process, can significantly improve its search performance. The techniques to find necessary assignments include static learning [1], dynamic learning [1], and recursive learning [2]. All these techniques did improve ATPG performance.

However, in our experience with real circuits, we found that necessary assignments can create unnecessary requirements in an ATPG process. Sometimes, these unnecessary requirements are not justifiable such that a testable fault may be mistaken as untestable.

2.0 Problem

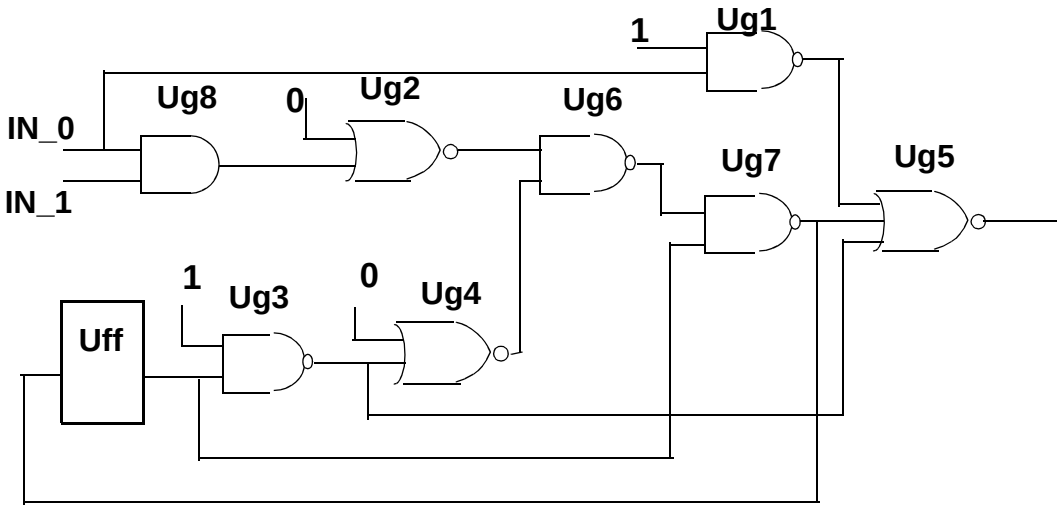

Figure 1.

In real circuits, due to performance or area consideration, there are usually some sequential elements which are not in scan chains. Figure 1 is a circuit with a nonscan flip-flop. Actually the circuit is bigger. However we remove the other portion which have constant values assigned after several decision steps. Here, only the Q output of the flip-flop Uff is used. Since every gate has single output only, in the following, the value of a gate means the value of the output of this gate. We can see that if Ug7 has value 0, then Uff has value 1, so as Ug4 and Ug6, then Ug2 will have value 0 and Ug8 will have value 1. Since Ug1, Ug3 and Ug7 have value 0, Ug5 must have value 1. Based on the contrapositive relationship used in static learning, dynamic learning or recursive learning, we will say if Ug5 has value 0, then Ug7 must have value 1.

To detect the stuck-at-1 fault at the output of Ug5, we need to put value 0 at the output of Ug5 which just need to set primary input IN_0 to have value 0. However, with the necessary assignments we learned, we will assign Ug7 to have value 1which in turn makes this fault untestable, as explained in the following.

2.1 Combinational ATPG

For a combinational ATPG, this nonscan flip-flop Uff will be treated as TIE-X gate which can generate unknown value only. To get Ug7 to have value 1, we need either Uff to have value 0 or Ug6 to have value 0. Since Uff is a TIE-X gate, the only chance is to have value 0 at Ug6 which in turn requires value 1 at Ug4. Due to Uff, Ug4 cannot have value 1 either. Therefore, we can conclude that there is no way to justify Ug7 to

have value 1. Therefore, we will declare the stuck-at-0 fault at the output of Ug5 as untestable.

It should be noted that, with symbolic simulation, Ug7 does have value 1 when Ug2 has a value 1. However, conventional simulation techniques will not see that.

2.2 Sequential ATPG

For a sequential ATPG, to justify value 1 at the output of Ug7, we can either set Uff to have value 0 or Ug6 to have value 0 which in turn will require Uff to have value 1. Therefore, we either need to set Uff to have value 0 or 1. If we choose 1, then in the previous cycle, we need to set Ug7 to have value 1 which means that the original requirement to have value 1 at Ug7 is repeated again. The only chance to get out of this repeated requirement, we have to choose the second choice which is to set Uff to have value 0. This will require that the previous cycle, Ug7 needs to have value 0 which in turn will require Uff to have value 1 again.

Since the requirements are repeated for every cycle, a sequential ATPG will treat this situation as not justifiable and declare this fault as untestable.

2.3 Another Example

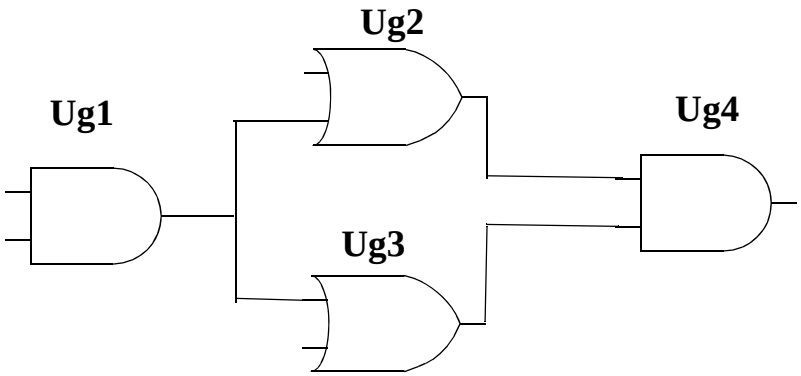

Figure 2

The first reaction after we found this problem is not to justify these derived necessary assignments. However, it does not work for the example in Figure 2. In Figure 2, we can see that value 1 at Ug1 makes value 1 at Ug4. In other words, value 0 at Ug4 requires value 0 at Ug1. Without justifying value 0 at Ug1, value 0 at Ug4 cannot be justified.

These two examples tell us that, to properly use necessary assignments in real circuits, we need to have better mechanism to tell which values assigned need to be justified.

3.0 Solution

In Figure 1, we see value 0 at Ug7 makes value 0 at Ug5. Based on contrapositive relation, value 1 at Ug5, requires that Ug7 cannot have value 1 (not-1). In binary world, we can say not-0 is 1, and not-1 is 0. However, in real circuits, due to the presence of X (unknown) values which can be created by nonscan flip-flops, and Z (high impedance) values which can be created by tri-state gates, not-0 can be 1, Z or X. In ATPG process, it is not necessary to justify X value; while values 0, 1 and Z have to be justified. In Figure 1, even Ug5 and Ug7 are binary gates, due to X-resolution problems of conventional simulation, they still can have X value.

We implemented not-0 and not-1 value assignments in our ATPG tools to solve this problem. For performance reason, we use not-0 and not-1 value assignments only at the area which can be affected by X value generators or Z value generators. X value generators include Tie-X gates, nonscan flip-flops. Z value generators include tri-state gates and bus gates. For other area, we still can treat not-0 as 1 and not-1 as 0. In the workshop, we will present our experimental data to validate our solutions.

4.0 References

[1] M. H. Schultz et al., “SOCRATES: A Highly Efficient Automatic Test Pattern Generation System”, IEEE Transactions on Computer-Aided Design, vol. CAD-7, No.1, Jan. 1988, pp. 126-137.

[2] Wolfgang Kunz and Dhiraj K. Pradhan, “Recursive Learning: A New Implication Technique for Efficient Solutions to CAD-problem: Test, Verification and Optimization”, IEEE Transactions on Computer-Aided Design, Vol. 13, No. 9, 1994, pp. 1143-1158.