False Path Search Tool Kit

Fan Zhang

fanzhang@uni-bremen.de

ITEM - University of Bremen – Germany

http://www.item.uni-bremen.de

Abstract

The False Path Search (FPS) tool kit under Linux / Solaris is a total solution for the false path problem in Static Timing Analysis (STA), based on the PDC library designed by Infineon Technologies.

1. Introduction

The initiative of developing the FPS tool kit is to solve the false path problem in STA, which means to check the consistency of the critical paths. A critical path is defined as a path that either violates or almost violates the given design constraints. Due to the local view of the STA procedure so called false paths could be reported as critical paths. A false path is a path that can never be sensitized under any operating conditions.
Given a critical path description with the values of the nets in the critical path and their side inputs (as well as side outputs for multi-output-gates) specified, the false path problem can be viewed as a consistency problem. If the values of the nets in the critical path as well as the side inputs are inconsistent considering the connectivity of the netlist, the critical path is called a false path.
Furthermore, based on the solution procedure, consistency is only checked for critical paths found by STA, thus timing information will not be be taken into account in the whole process. This fact in conjunction with efficient algorithms dramatically improves overall efficiency.

2. General solution flow

A simple example is used here to demonstrate the problem and general solution flow, as shown in Fig. 1. It is a short circuit with a critical path (in red) as well as its side inputs (in green), which are already assigned logic values.

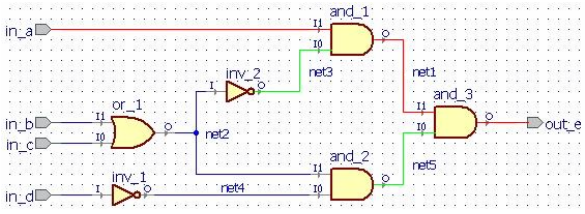

Figure 1: A simple circuit with a false path

Therefore, the critical path for this example is: in_a => net1 => out_e, and the side nets are thus net3 and net5. They form the critical path nets together, and have their values when the critical path is generated: in_a = f, net1 = f, out_e = f, net3 = 1, net5 = 1. The nets in the critical paths are assigned with edge values: f (falling edge, equal to 0) and r (rising edge, equal to 1), while the side nets have their non-controlling values for their neighbours in the critical paths.
The inconsistency can easily be found after the simple implication: net2 should be 0 because of the gate inv_2 (net3 = 1), but it should also be 1 because of the gate and_2 (net5 = 1). Thus, the conflict happens on net2.
According to the definition of the false path problem, the value assignments should cover all the nets in a circuit to check the consistency. However, only the nets that form reconvergences with the critical path (such as net2 in the example) influence the consistency and must be considered. Hence, the general solution flow would be:
- initialize the data-container for saving test data for the critical path nets
- check consistency of the initial values
- traverse circuit by importing nets; if an indicator for a reconvergence appears, search the reconvergence
- make implications for the reconvergent nets
- check consistency.

Moreover, in the circuit traversal part, two graph algorithms (BFS and DFS) are integrated; supplementary, two net importing methods (all directions importing and backward importing) are also developed.
At last, some important parameters are added to make the program much faster and more robust. For instance, runtime per critical path should be controlled, and the number of possible assignments is also critical.

3. Applying FPS on an ISCAS benchmark

The ISCAS benchmark C6288 is used for the demonstration of using FPS tool kit. A part of the traversed circuit using BFS, reconvergence searching and false path check results, are shown in the following figures. However, the animation of circuit traversal is not visual here.
The key parameters of C6288 are shown in Tab. 1. The circuit diagram in Fig. 2 is generated with BFS algorithm (using all directions importing method).

Model name	C6288
Number of inputs	32
Number of outputs	32
Number of total gates	2416
Number of generated critical paths	64

Table 1: Key parameters of C6288

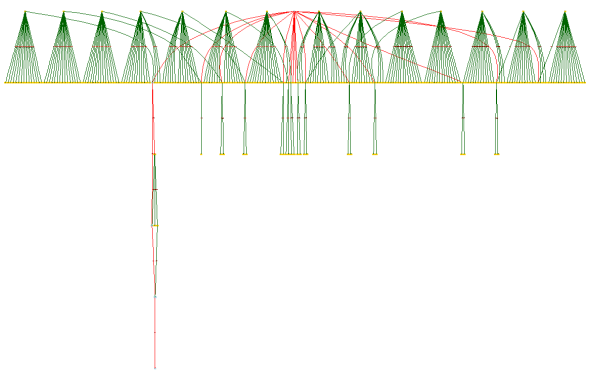

Figure 2: A part of traversed circuit C6288 using BFS

The reconvergence searching using BFS is extreme fast: the maximum number of reconvergences is 2300, and the elapsed time is only 2.2 seconds per path. However, the speed of DFS is slower: for the same number of reconvergences, it takes 8.4 seconds (see Fig. 3-4). Hence, it is advantageous to use BFS for the consistency checking.

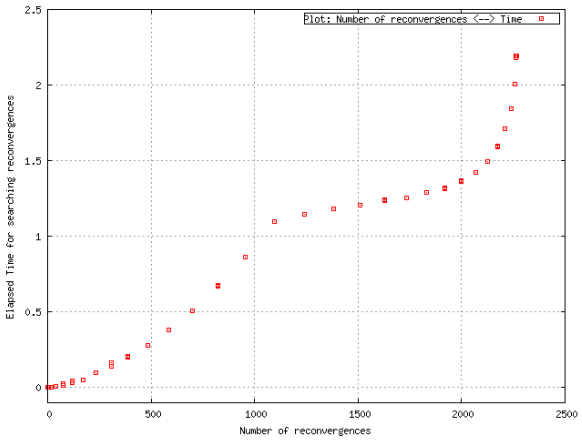

Figure 3: Reconvergence searching using BFS

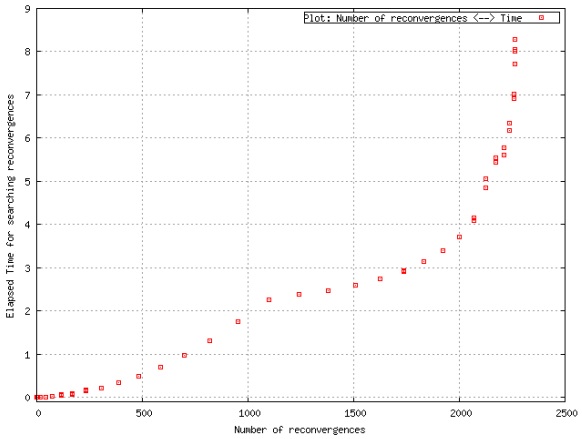

Figure 4: Reconvergence searching using DFS

While verifying critical paths using FPS, the following parameters are applied: 100 seconds (maximum runtime per critical path), 1 (limit of possible assignment combinations during reconvergence searching) and 30 (limit of total possible assignment combinations). Less than 1 second is needed for false paths; nevertheless, most true paths need a comparative long time to accomplish, because the program is trying to find out up to 30 combinations per path after reconvergence searching (see Fig. 5, square is true path, consistency is 1; bubble is false path, consistency is 0).

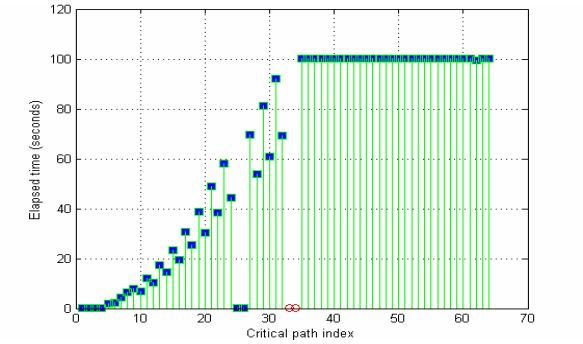

Figure 5: False path test of C6288 based on BFS

4. Conclusion

In this project, we have achieved to solve the false path problem in a perfect way, and additional functions, such as circuit diagram generation, circuit traversal animation and data plotting, are also added into the tool kit to make the algorithm check and result verification much easier and more convenient. Besides, some heuristic parameters are designed to speed up the total process, though the procedure itself is non-heuristic.

5. References

[1] F. Zhang, “Search Algorithms for False Paths in Static Timing Analysis”, diploma thesis in ITEM, University of Bremen, October 2006.
[2] A. Kuehlmann, M. Ganai and V. Paruthi, “Circuit-based Boolean Reasoning”, DAC 2001, pp. 232-237.
[3] D. Blaauw, R. Panda and A. Das, “Removing User-specified False Paths from Timing Graphs”, DAC 2000, pp. 270-273.

Acknowledgement

The author would like to thank Mr. Bargfrede (Infineon Technologies) and Prof. Anheier (ITEM) for their dedicated guidance and passional help during the whole project. Besides, special thanks are given to Infineon Technologies AG for providing the PDC library and offering abundant outlays for this work.