# Path Verification Using Boolean Satisfiability

Matthias Ringe, Thomas Lindenkreuz
Robert Bosch GmbH, 72703 Reutlingen, Germany

Erich Barke
Institute of Microelectronic Systems, University of Hanover, 30167 Hanover, Germany

**Abstract**

*The importance of identifying false paths in a combinational circuit cannot be overstated since they may mask the true delay. We present a fast algorithm based on boolean satisfiability for solving this problem. We also present extensions to this per-path approach to find the critical path of a circuit in a reasonable time.*

## 1 Introduction

Static timing analysis provides an efficient method to find the delay of a combinational circuit. The circuit is translated into a directed acyclic graph with delays for edges and vertices. However, the possible existence of false paths may lead to an overestimation of the delay. To overcome this restriction, the sensitization of paths has to be proven. This is known as the general false path problem.

The general false path problem implies two special problems: 1. The definition of a sensitization criterion. 2. The algorithmic aspects of verifying the criterion. The viability-criterion [1][2] and its slight modifications [3][4][5] are widely accepted. Since we concentrate on the algorithm, we also use this criterion. Previous approaches are based on the D-Algorithm (e.g. [3][4][6]), BDDs [1][7] or boolean satisfiability [7]. The integration of searching the longest path and simultaneously testing the sensitization criteria is known as online-processing. Recent work concentrated on this problem [3][4].

We use the definitions and notations presented in [2], [4] and [10].

## 2 Verifying a single path

We use the following sensitization criterion (see also [1][2]): At each gate of the path all early-arrive signals (i.e. signals that are stabilized before the current event) must present the non-control value. If there are late-arrive signals, these are ignored which implicitly assumes that they have the non-control value.

Using boolean satisfiability is a very efficient method for automatic test pattern generation which has been demonstrated in [9]. The circuit is represented by a conjunctive normal form (cnf) formula. See [9] on how to obtain this formula for a given circuit. The formula can be built in linear time with respect to the number of gates. Trivially, this formula is satisfiable. A contradiction can be created by the sensitization conditions which are concatenated (boolean and-operation) with the cnf formula. If this formula is not satisfiable, the current path is a false one.

We use a program called "POSIT" written by J. W. Freeman [10] to examine the satisfiability of the sensitization formula.

Although POSIT is very efficient, its runtime can explode in an exponential way with respect to the number of clauses and propositions. Since the satisfiability check is usually very fast, we iterate building the formula. Every predecessor of the current path gets a value describing the minimal topological distance (number of gates) to the path (mdtp). The formula is built regarding only the vertices which have a mdtp less or equal than a given bound. The bound is increased until the path is identified as false or all predecessors are reached and the path is true.

## 3 Searching for the critical path

It is known for some time that the per-path approach is not sufficient for finding the critical path in a circuit. The online-processing algorithm of [6] uses esperance, which is the greatest possible delay from the current edge to a primary output, to iteratively build and check sub-paths to find the critical path in a circuit. We extend this approach to a technique we call dynamic esperance.

Let in Fig. 1 path $P_1$={*A,D,E,G,H*} be the one first generated. All gates present a unit delay of one. The interconnect does not have a delay. The numbers on the interconnect show the esperance. The path $P_1$ is obviously false, since *F* cannot present a non-control value for the gates *U4* (a boolean "1") and *U5* ("0"). All paths which contain the sub-path $P_S$={*E,G,H*} and reach *E* after time 1 are false paths. Therefore, the esperance at *E* after time 1 is 1, at *D*

it is 2 and at *A* and *B* it is 3 (see values in round brackets in Fig. 1).

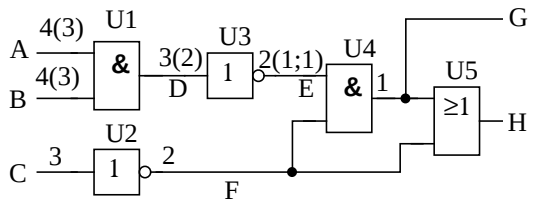


Fig. 1: Dynamic Esperance

The satisfiability checker is used to find those input pins $i_0$, ..., $i_n$ (*U4/E*, *U5/G* in Fig. 1) that are necessary for a path to be false. The algorithm consists of the following steps:

1. Calculate a time threshold $T_{th}$ for pin $i_0$. All paths through $i_0$, ..., $i_n$ whose events reach $i_0$ after $T_{th}$ are false. $T_{th}$ is obtained by a breadth-first search. In Fig. 1 $T_{th}$ is 1 at pin *E* at gate *U4*.
2. Mark all edges through $i_0$, ..., $i_n$.
3. Save the esperance for all marked edges.
4. Set the esperance at $i_n$ to -∞.
5. Backpropagate the dynamic esperance values for all marked edges. As for the normal esperance calculation, only the maximum esperance values are backpropagated. In Fig. 1: Output *G* of *U4* gets an esperance of 0, *E* gets 1.
6. Restore the esperance for all edges for all paths through $i_0$, ..., $i_n$. Compare the dynamic esperance at $i_0$ with the static one. If it is smaller, backpropagate it under the following rules to the primary inputs: The dynamic esperance data structure contains the esperance value and a time threshold for which the esperance is valid (Notation (1;1) in Fig. 1: the esperance of 1 on interconnect *E* is valid after time 1). Only the maximum esperance without respecting the time threshold is backpropagated.

Note that the dynamic esperance is only valid for $i_0$ and all its predecessors. The dynamic esperance becomes the static esperance if the threshold value falls below the shortest path from a primary input to the current edge (min-delay-from-source). This is the case for the edges *A*, *B*, *D* in Fig. 1.

This path blocking technique prevents the generation and testing of long false paths.

## 4 Experimental Results

The program was implemented in C++. The publicly available ISCAS-85 benchmarks were run on a Sun Ultrasparc 1 workstation. Our results are summarized in the following table (CPU time without parsing the netlist and building the graph).

| Circuit | Longest topol. path | Critical path | Time to find (CPU sec.) |
|---|---|---|---|
| C880 | 24 | 24 | <1 |
| C1355 | 24 | 24 | 3 |
| C1908 | 40 | 37 | 8 |
| C2670 | 32 | 31 | 27 |
| C3540 | 47 | 46 | 21 |
| C5315 | 49 | 47 | 21 |
| C6288 | 124 | 124 | 5 |
| C7552 | 43 | 42 | 15 |

## 5 Conclusions

The problem of verifying a single path and finding the critical path in a combinational circuit has been considered. We presented a fast algorithm based on boolean satisfiability using the program POSIT written by J. W. Freeman and complexity reduction techniques. The dynamic recalculation of the esperance was introduced to allow to search for the critical path. These techniques are efficient and practical.

## References


[1] P. C. McGeer, R. K. Brayton, “Efficient Algorithms for Computing the Longest Viable Path in a Combinational Network”, *Proc. of the 26th ACM/IEEE DAC*, pp. 561-567, 1989

[2] D. H. C. Du, S. H. C. Yen, S. Ghanta, “On the General False Path Problem in Timing Analysis”, *Proc. of the 26th ACM/IEEE DAC*, pp. 555-560, 1989

[3] H. Chang, J. A. Abraham, “VIPER: An Efficient Vigorously Sensitizable Path Extractor”, *Proc. of the 30th ACM/IEEE DAC*, pp. 112-117, 1993

[4] S. Devadas, K. Keutzer, S. Malik, “Computation of Floating Mode Delay in Combinational Circuits: Theory and Algorithms”, *IEEE Trans. on Computer-Aided Design*, vol. 12, no. 12, pp. 1913-1923, December 1993

[5] K. Keutzer, S. Malik, A. Saldanha, “Is Redundancy Necessary to Reduce Delay?”, *IEEE Trans. on Computer-Aided Design*, vol. 10, no. 4, pp. 427-435, April 1991

[6] J. Benkoski, E. V. Meersch, L. Claesen, H. De Man, “Efficient Algorithms for Solving the False Path Problem in Timing Verification”, *Proc. of the ICCAD*, pp. 44-47, 1987

[7] P. C. McGeer, R. K. Brayton, “Integrating Functional And Temporal Domains in Logic Design”, *Kluwer Academic Publishers*, 1991

[8] E. V. Meersch, L. Claesen, H. De Man, “Static Timing Analysis of Dynamically Sensitizable Paths”, *Proc. of the 26th ACM/IEEE DAC*, pp. 568-573, 1989

[9] T. Larrabee, “Test Pattern Generation Using Boolean Satisfiability”, *IEEE Trans. on Computer-Aided Design*, vol. 11, no. 1, January 1992

[10] J. W. Freeman, “Improvements to propositional satisfiability search algorithms”, *Ph.D. thesis, University of Pennsylvania*, 1995