Timing Analysis with known False Sub Graphs

Krishna P. Belkhale *
AMBIT, Sunnyvale, CA 94086
belkhale@ambit.com

Alexander J. Suess
IBM, East Fishkill, New York 12533
ajsuess@vnet.ibm.com

Abstract *- In this paper we formulate the problem of timing analysis with known false sub graphs. This problem is important when we want the timing analysis system to take into account false path information that is supplied either by the user or by another program, and supply accurate timing information to optimization programs such as placement and wiring. We present an efficient algorithm for the problem.*

Introduction

Static timing analysis algorithm, as described in [1], is currently a widely used mechanism for efficiently verifying the timing behavior of circuits.. The algorithm computes the arrival time of signals in a forward pass through the timing graph. The second step involves propagating the required times computed at the latches in a backward pass through the graph. At the end of two passes each vertex in the timing graph has a computed arrival time *AT* and required arrival time *RAT*. The quantity slack *SLK* is then computed as *RAT - AT*. This gives a good local measure of the magnitude of the timing violation.

The problem with the static timing analysis procedure described above is that it does not take logic into account. Thus some of the paths that are considered by the algorithm may not be logically realizable. These paths are often referred to as false paths. Such paths must be detected and eliminated from consideration from the timing analysis. This problem has been studied extensively by many researchers, and various interesting algorithms for false path detection and elimination have been discovered [2,3,4,5,6,7].

In this paper, we formulate and analyze a new problem of timing analysis given a set of false sub graphs. The notion of false sub graphs is more general than the notion of false paths as we can simultaneously remove the consideration of multiple paths.

This problem is useful for a variety of reasons. In many cases, users do have an idea that certain paths reported by the timing system are really false. This formulation allows the users to convey this information to the timing system resulting in a more meaningful analysis. As will be shown in Section 2, the ability to remove entire sub graphs from consideration from timing is a powerful feature.

A second reason for looking into this problem is that though techniques described in [2,3,4,5,6,7] solve the problem of determining the true delay of the circuit, they do not determine slacks at all points in the design. These slacks are a useful input to physical optimization programs, such as placement and wiring. Our problem formulation and solution, with the false sub graphs as input, allows us to still maintain the notion of slacks in the design. The false sub graphs may either be provided by the user or determined automatically. An interesting problem arises in this context. The determination that a given path is false depends on both logic and temporal factors. Thus, a false path may not stay false through a physical optimization process. For a common definition of a false path, the loose criterion defined in [4], one could prove that removing the false paths from consideration during a series of physical optimization steps (note the false paths were determined at the start of optimization) will only result in an analysis that does not underestimate the delays in the circuit. Thus, the result of the analysis is still useful and the actual false paths need not be re-determined during the optimization process.

The outline of the paper is as follows. In Section 2, we formulate the problem of timing analysis with known false sub graphs. In Section 3, we describe an algorithm for solving the problem. In Section 4, we describe some implementation results. In Section 5, we give the conclusions.

2 Problem specification

In this section, we formulate the problem of timing analysis with known false sub graphs. For our discussion, we assume that the timing model of a circuit is a graph. The graph is assumed to be acyclic. The edges of the graph are associated with the delays.

Let $F_1,..., F_k$ be k false sub graphs, which are sub graphs of G. We define the sets B_i, E_i, $1 \leq i \leq k$, which are referred to as the *begin set* and *end set* respectively, as follows.

Definition 1 *The begin set B_i is the set of vertices x in each sub graph F_i such that there are edges in the sub graph that start from x but there are none that end in x. Similarly, the end set E_i is the set of vertices x in each sub graph F_i such that there are edges in the sub graph that end in x but none that start from x.*

* The work was done while the author was at IBM

The problem considered in this paper can be defined as follows. An example is illustrated in Figure 1 .

Definition 2 *The problem of timing the graph G with false sub graphs $F_1,...,F_k$, consists of computing the timing quantities AT, RAT and SLK for all the vertices in G, while disregarding some paths from the analysis. The paths to be disregarded are the set of all paths in G, which have a sub path in some F_i, that starts at a vertex in B_i and ends in a vertex in E_i.*

In the definition above, if the false sub graphs are specified by the user, then there is a necessity for making the user specification easier. One mechanism involves specifying F_i as a set of ordered pair of vertices. An ordered pair (v_l, v_m) implies the inclusion of all the edges and vertices in G, that lie on a path from v_l to v_m, into F_i. The sub graph F_i can be specified as a collection of such ordered pairs. Note that this specification in some cases may be the same as specifying all the edges in F_i, resulting in no compression in specification. For example in Figure 1, F_1 cannot be specified as $\{(v_1, v_7)\}$ as this will result in the inclusion of the diagonal edges. Instead, it needs to be specified as a set of seven ordered pairs corresponding to the seven edges of the graph. However, in many cases this will allow compact user specification of false sub graphs. For example, consider the standard false path situation as shown in Figure 2. Assuming the control path delays are small, all paths leading from the I_1 pin of the first multiplexer MUX_1 to the I_1 pin of the second multiplexer MUX_2 are false. This is because the control signal setting that allows the propagation at the first pin will block the propagation at the second pin. This situation can be specified by using a single ordered pair. The false sub graph is represented by the set $\{(MUX_1/I_1, MUX_2/I_1)\}$.

3 Algorithm for the problem

The overall approach of the algorithm is the same as described in [1], except that the timing information computed at a node is now more complex. The algorithm computes multiple arrival and required times at a node. The different arrival and required times at a node are distinguished based on a set attribute. The set attribute is a subset of the set $\{1,...,k\}$, where k is the number of input false sub graphs. In other words, the set attribute is an element of the power set of $\{1,...,k\}$. For example, with k = 2, the possible values for the set attribute are $\{\}$, $\{1\}$, $\{2\}$, and $\{1, 2\}$. The set attribute value gives the set of false sub graphs the signal has come through.

At a node in the graph, some of the elements of the power set are associated with timing information. The actual elements that have associated timing information at a node is determined by the algorithm, which is described below. We will use the notation $AT(v, s)$, $RAT(v, s)$, $SLK(v, s)$ to denote the respective timing quantities at node v for the element s of the power set.

We first describe the arrival time propagation phase of the algorithm. The required time propagation is essentially the inverse of the arrival time propagation. Once these are calculated at a node v, the slack is obtained by first computing $SLK(v,s) = RAT(v,s) - AT(v,s)$ for all s, where s is an element of the power set with a valid time at the node. The slack at the node is then computed as the minimum of the quantities $SLK(v,s)$ for all elements s of the power set at the node.

The arrival time propagation phase is a breadth first approach starting at the primary inputs. The arrival time at the primary inputs are based on user assertions. These times are taken to be associated with the null set $\emptyset$. A node v is processed for arrival time computation only if all the edges incident to v come from vertices whose arrival time information has already been computed. Since the timing graph G is acyclic, this algorithm will result in the computation of arrival times at all nodes in the graph. The arrival time processing at node v is described below, following some definitions.

Definition 3 *For every node v in G, we define the set BG(v) to be the set of all false sub graphs for which the vertex v is a element in the begin set. In other words, $BG(v) = \{i | v \in B_i\}$. Similarly, we define EG(v) to be the set of all false sub graphs for which the vertex v is in the end set. In other words, $EG(v) = \{i | v \in E_i\}$.*

Definition 4 *For every edge e in G, we define the set IN (e) to be the set of all indices of false sub graphs that contain e.*

Algorithm 1

```
---------------------------------------------
arrival_process ( v ) {
  for each edge e ← (u, v) {
     for each element s ( in power set )
           with valid time at u do {
        s' ← (s ∪ BG (u)) ∩ IN (e)
        if ( s' ∩ EG (v) = ∅ {
            AT(v, s') ← max(AT(u,s) + delay(e), AT (v,s'))
          }
        }
     }
  }
---------------------------------------------
```

An element set indicates the set of false sub graphs the arriving signal has propagated through. The algorithm above takes each arrival time at the source of an edge and derives the sink arrival time and the sink element set. The sink element set is obtained by taking the source element set and performing a union with *BG (source)* , followed by intersection with *IN (edge)*. The union operation signifies the new false sub graphs that begin at the source point. The intersection operation signifies that the sink element set cannot contain indices of false sub graphs that do not contain the edge e. If the sink element set contains an index of a false sub graph that ends at vertex v, then the propagated arrival time is ignored. This condition is checked by determining if s' and *EG (sink)* have a valid intersection. If they do not, then the arrival time at the sink is updated to the maximum
of the computed arrival time and the previously computed result (which is taken as $-\infty$ if there is none before). The required arrival time computation proceeds in a reverse manner to the arrival time propagation. The complete algorithm result for the example in Figure 1 is shown in Figure 3.

The performance of the algorithm depends heavily on the number of elements of the power set that are associated with valid times at a node. In the most general case, the number of elements at a node v is bounded above by 2^f, where f is the number of false sub graphs that contain v. This bound will be reached only when there are varied patterns of intersections between the false sub graphs. If all the false graphs are really just false paths, we can prove the following theorem.

Theorem 1 *If all the false sub graphs are just false paths, then the number of elements with valid times at a node v is at most f, where f is the number of false paths that go through v.*

Proof: Let s be an element with a valid time at node v. Among all the false path indices in set s, let i be the index of a false path whose sub path sub_i, till node v is the longest. It can be seen that the sub paths of the other false paths in set s till node v are also sub paths of sub_i. Also, if a false path, with index x, has a sub path starting from its start vertex till node v which is a sub path of sub_i, then the index x must be an element of set s. Based on the above observations the set s is totally determined by the index of the false path with the maximum sub path. Since there are only f choices for this index, there can be only f different set elements at the node.

It should be noted that the above theorem does not imply that it is advantageous to deal only with false paths. This is because the number of false paths that are needed to replace a false sub graph may be exponential in terms of the size of the false sub graph.

4 Implementation Results

The algorithm for timing analysis with known false sub graphs has been implemented as part of a timing analysis program. The performance results of the algorithm are presented for a two dimensional array of blocks as shown in Figure 4. Each of the blocks in the circuit has two inputs and two outputs. The timing graph for a block connects each of the inputs to the outputs by direct edges. The results presented in this section are for a mesh of size 100, with number of edges in the timing graph equal to 60,200.

Total False Edges Paths	False Per Path	Avg. Edge	Time
0	0	0.0	12.6
30,000	159	188.7	20.1
60,000	314	191.1	29.8
90,000	463	194.4	39.3
120,000	609	197.0	51.4
150,000	760	197.4	60.9
180,000	919	195.9	74.2
210,000	1063	197.6	86.4
240,000	1218	197.0	107.5
270,000	1357	199.0	123.8

Table 1: Performance for false paths on 100 by 100 mesh

In Table 1, we give the execution time for timing analysis for varying number of false paths. The times are in seconds on a IBM RS/6000 processor. The false paths were randomly generated by walking backwards randomly from outputs of the mesh to the inputs of the mesh. The first column gives the total number of false path edges that were asserted. This is just the count of the total number of edges in each false path. The second column gives the number of false paths. The third column gives the average number of edges in a false path, which can be derived from the first two columns. The fourth column gives the execution time of the timing analysis algorithm. The table shows the performance of the algorithm from 0 false path edges to 270,000 false path edges (which is more than 4 times the original timing graph size). The table shows that the performance is fairly linear in the range.

Total False Edges Paths	False Sub Graphs	Avg. Edges Per Sub Graph	Time
0	0	0.0	12.6
30,000	29	1034.5	19.3
60,000	54	1111.1	29.6
90,000	80	1125.0	43.2
120,000	107	1121.5	52.6
150,000	133	1127.8	65.3
180,000	153	1176.5	73.2
210,000	169	1242.6	87.0
240,000	194	1237.1	103.6
270,000	217	1244.2	120.0

Table 2: Performance for false sub graphs on 100 by 100 mesh

In Table 2, we give the execution time for timing analysis for varying number of false sub graphs. The false sub graphs were randomly generated by walking backwards randomly from outputs of the mesh to the inputs of the mesh. Instead of going backwards on only one edge to a point, a secondary edge was also traced back on certain points. These points were chosen randomly based on an input probability. The first column gives the total number of false path edges that were asserted. This is the total of the number of edges in the individual false sub graphs. The second column gives the number of false sub graphs. The third column gives the average number of edges in a false sub graph, that can be derived from the first two columns. The fourth column gives the execution time of the timing analysis algorithm. The table shows the performance of the algorithm for the identical range of false path edges as in Table 1. The overall performance matches the performance shown in Table 1. It should be noted that although the two tables contain similar results for matching numbers of false edges, the sub graph results are more impressive because they suppress many more false paths. This is because a sub graph is, in general, equivalent to many individual paths, but a sub graph specification uses far fewer edges.

5 Conclusions

We formulated the problem of timing analysis with known false sub graphs and presented an algorithm for solving the problem. Furthermore, we described some performance results of the implementation, which showed the effectiveness of the problem formulation and solution.

References

[1] R. B. Hitchcock, "Timing Verification and Timing Analysis Program", *Proc. of the 19th ACM/IEEE DAC* 1982, pp. 594-604.

[2] D. H. C. Du, S. H. C. Yen, and S. Ghanta, "On the General False Path Problem in Timing Analysis", *Proc. of the 26th Design Automation Conference* 1989, pp. 555-560.

[3] P. McGeer and R. K. Brayton, "Efficient algorithms for computing the longest viable path in a combinational network", *Proc. of the 26th Design Automation Conference* 1989, pp. 561-567.

[4] H. C. Chen, D. H. C. Du, "Path Sensitization in Critical Path Problem", *IEEE Trans. on CAD* Feb. 1993, pp. 196-207.

[5] S. T. Huang, T. M. Parng, and J. M. Shyu, "A Polynomial-Time Heuristic Approach to Approximate a Solution to the False Path Problem", *Proc. of the 30th Design Automation Conference* 1993, pp. 118-122.

[6] H. Chang, and J. A. Abraham, "VIPER: An Efficient Vigorously Sensitizable Path Extractor", *Proc. of the 30th Design Automation Conference* 1993, pp. 112-117.

[7] P. McGeer and R. K. Brayton, "Integrating Functional and Temporal Domains in Logic Design", Norwell, MA: Kluwer Academic, 1991.

Acknowledgements:

The authors acknowledge valuable discussions with Mr. Jitendra Apte, Mr. Rajesh Gupta and Mr. Deepak Sherlekar on the applicability of the problem in physical design.

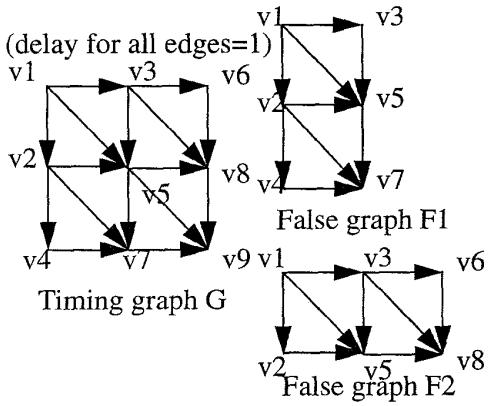

Fig. 1. Timing graph and false sub graphs.

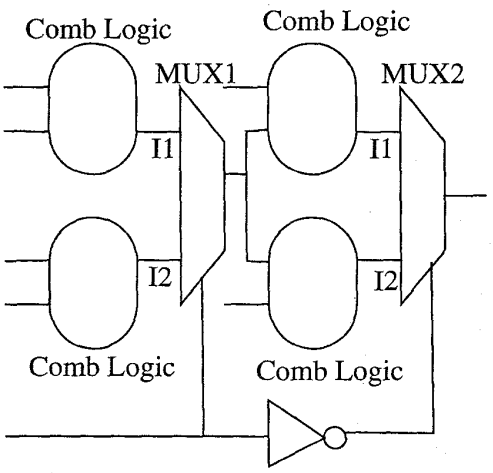

Fig. 2. A standard false path test case.

V1	V2	V3	V4	V5	V6	V7	V8	V9
(0,{})	(1,{1,2})	(1,{1,2})	(2,{1})	(1,{}) (2,{1,2})	(2,{2})	(2,{})	(2,{})	(3,{})
(2,{})	(3,{1,2})	(3,{1,2})	(N,{1})	(3,{}) (4,{1,2})	(N,{2})	(4,{})	(4,{})	(5,{})

Fig. 3: Arrival Time and Required Time propagation (N = No RAT)

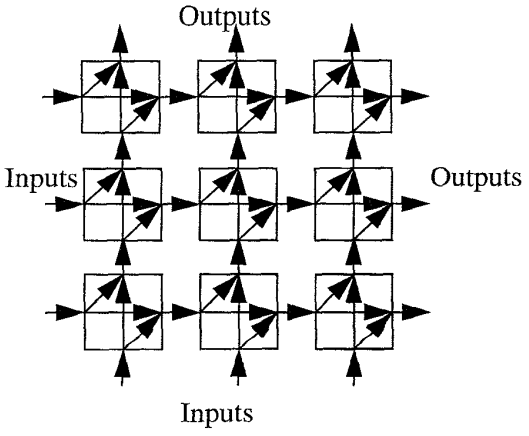

Fig. 4. Two dimensional Mesh.