

On False Timing Paths and Dynamic Slack

by

B. Könemann

LogicVision, Inc., San Jose, CA

T. W. Williams

IBM, Corp., Boulder, CO

Static Timing Analysis is a very powerful and increasingly popular tool for complex designs. The slack values produced by the analysis are extremely valuable for guiding synthesis and layout. However, static timing analysis can be pessimistic by including so called false paths that cannot be functionally sensitized. To improve the accuracy of the analysis, it is desirable to identify false paths and remove them from consideration. False path identification requires dynamic analysis of signal switching activities including possible glitches. It will be demonstrated by example how dynamic effects lead to significant complications in the definition and calculation of updated slack values after false path identification and removal.

Motivation

The work presented here was originally stimulated by a paper [1] which proposed to use static path sensitization techniques for false path identification. At the time, we derived a simple dynamic path sensitization method to demonstrate the inadequacy of static path sensitization in the presence of possible glitches. Other authors quickly came up with various techniques for more accurate false path identification and removal [3-5] which avoid the trap of static path sensitization. The reason for revisiting this issue now is to take the discussion one step further by beginning to investigate how false path removal impacts the calculation of slack values.

1. Background

Before we begin the discussion of dynamic effects in slack calculations, we briefly review some basic concepts.

Static timing analysis [2] in essence calculates a local signal Arrival Time (AT) for each block in a structural network based on topological delays associated with the blocks and nets. For the following discussion we make the simplifying assumption that net delays can be lumped into the block delays. The AT values are calculated in a forward pass beginning at the network inputs, starting from user-provided assertions for each input AT. The signal AT at block outputs is the worst case (earliest or latest) sum of each block input AT

and the worst case (shortest or longest) delay between the respective block input and block output.

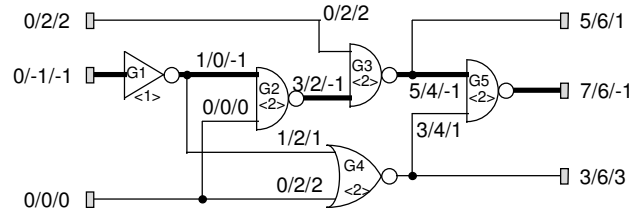


Figure 1: Static Timing Analysis Example

A simple example of a long path analysis is illustrated in Figure 1. It is assumed that the inverter delay is 1 time unit, while the other gates have a delay of 2 time units. Further, it is assumed that the all network input AT values are 0; that is, no network input signal is allowed to switch after time 0. The leftmost of the three numbers ($x/y/z$) on each net denotes the corresponding local signal AT values in long path mode. For example, the upper input signal of G2 arrives at time 1, while the lower input signal arrives at time 0. Long path analysis derives the block output AT by adding the block delay to the each block input AT and picking the latest result, which is 3 for G2 (the latest of 0+2 and 1+2). Similar calculations are used on the other gates. The expected latest output from the network is available at time 7 (second network output from top).

The middle number on each net is a local Required Arrival Time (RAT), which represents the timing objective. The objective is asserted by the user in the form of RAT values for the network outputs. For the example it is assumed that the RAT is 6 for all outputs. The RAT values are pushed back from the outputs into the network by subtracting block delays along the way. That is, the RAT value at block inputs RAT value at the block output minus the block delay. For example, the RAT at the output of G5 is 6. The delay through G5 is 2. That is, to meet the required output arrival time, the gate input values must arrive no later than at time 4 (which is 6-2). At fan-out nodes, the RAT values from different fan-out branches must be consolidated into a single value for the fan-out stem. For long path analysis, the smallest of the individual RAT values from the branches is used for the fan-out stem.

The rightmost of the three numbers on each net is the Slack value. Slack is the difference between the local Required Arrival Time and the predicted local Arrival Time (i.e., $Slack = RAT - AT$). Positive slack indicates that the signal arrives earlier than required, and could be slowed down without causing timing problems. Negative slack, on the other hand, means that the signal violates the required arrival time and must be accelerated. The highlighted path in Figure 1 has negative slack and, therefore, fails to meet the required timing.

Slack information is very useful for many other applications like synthesis, or placement and wiring, to assure that no timing violations exist in the circuit, and that no power and area is wasted on signals that are faster than needed. Because of this utility, slack values probably are the most valuable output from timing analysis.

2. False Paths and Static Sensitization

Static timing analysis algorithms tend to be local, processing one block or one fan-out at a time. The local nature of the approach ignores considering cumulative functional effects which can make it impossible to propagate signal edges along certain paths. Such paths are called false paths. Without knowing which paths are false, static timing analysis results can be pessimistic by overestimating the latest (or underestimating the earliest) signal arrival times.

Reference [1] suggests to use traditional static test generation concepts for validating the propagation paths. Sensitizing a signal path through a circuit requires that all side-inputs to the logic gates along the path are set to the non-controlling value (logic '1' for AND/NAND, logic '0' for OR/NOR). It is well known from test generation that reconverging logic can make it impossible to simultaneously achieve all sensitization requirements for a path.

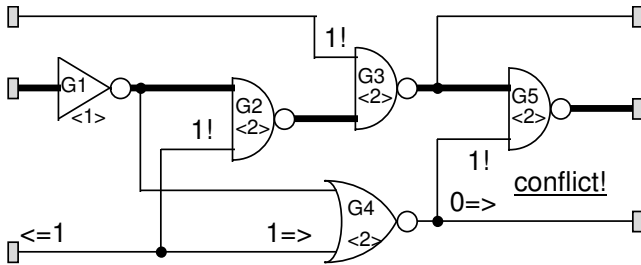


Figure 2: Static False Path Example

Figure 2 illustrates static sensitization analysis for the longest path (7 time units) through the example circuit from Figure 1. Some special notation is introduced. 1! and 0! denote logic values that are locally required for sensitizing the selected path. <=1 and <=0 represent

values derived from backward implication, while 1=> and 0=> are used to indicate forward implication. For example, the lower input of gate G2 must carry a 1! to sensitize the highlighted path. To accomplish the 1! implies a <=1 for the third network input, and a 1=> propagates forward to the bottom input of the NOR gate, G4. The 1=> forces the output of G4 to 0=>. This value is in conflict with the 1! required for extending the highlighted path through G5. The apparent conclusion is that the selected path cannot be sensitized and, therefore, is a false path.

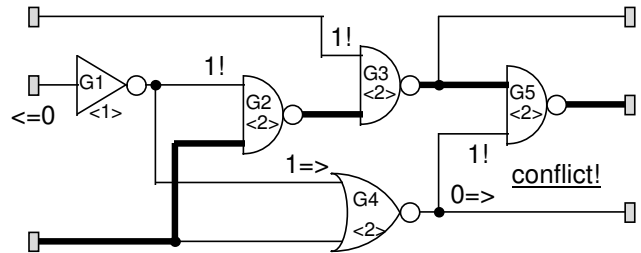


Figure 3: Another Static False Path

Figure 3 shows that the second-longest path (6 time units) to the second network output also cannot be statically sensitized. All other paths to this output are 5 time units or less. Hence, the static path sensitization analysis claims that no signal activity should be possible at the second network output after time 5.

3. Dynamic Path Sensitization Effects

While the static path argument looks compelling at first glance, it ignores dynamic signal effects. This observation is not new [3-5], and can easily be illustrated by drawing some wave forms for the example circuit.

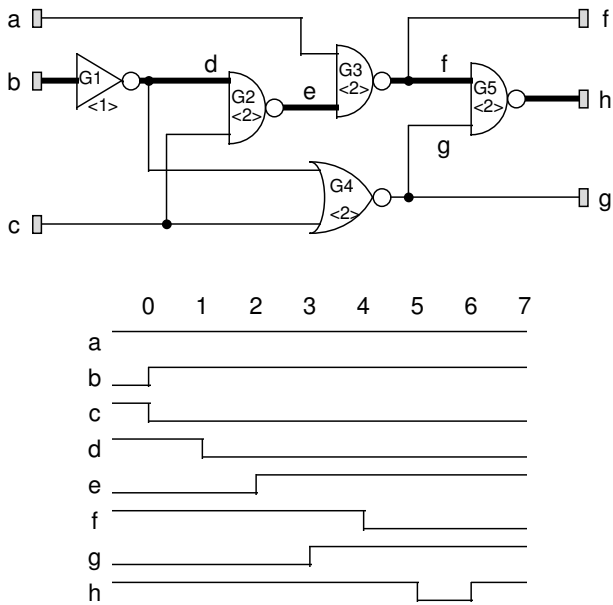


Figure 4: Wave Forms for Example Circuit

The wave form for net h in Figure 4 shows activity up until time 6. Yet, the static path analysis claims that the highlighted path with 6 time units cannot be sensitized. The wave forms prove that this clearly is misleading.

Where did the static path analysis fail?

Static path sensitization assumes that the gating conditions along the path are stable. However, the gating signal from the second network input in the wave form example is switching at time 0. As a result, the static conflict between the 0=> and the 1! at G5 in Figure 3 is resolved over time: it starts out at 0 and then switches to 1. This contributes to a glitch in the output signal. The static path sensitization technique fails to recognize the possibility of such glitches.

A simple dynamic sensitization approach is used in the following merely as a tool of discussion and to lay the foundation for the concept of dynamic timing sensitivity analysis. There is no claim that the method is novel, nor that it is a practical solution for the extremely hard problem of false path identification in large networks.

The method is based on the observation that temporary, rather than static, sensitization conditions are sufficient for propagating signal edges. This introduction of the parameter time necessitates some new notation for time-dependent signals. 1.t and 0.t indicate temporary value requirements (must be logic 1 or logic 0 at time t). As before, an exclamation mark, !, indicates values that are required for path sensitization on side-inputs of gates along the selected path, while => and <= denote forward and backward implication. Two ad-

ditional value types are introduced to represent the constraint that network inputs cannot switch after time 0. The notations 1.t+ and 0.t+ (meaning the signal is at a stable logic '1' or logic '0' for times later than t) describe this situation. Stable values are of unique importance, because stable value conflicts cannot be resolved. That is, simultaneous values of 1.t1+ and 0.t2+ on a single net are not possible. By contrast, non-stable signal values 0.t1 and 1.t2 can be resolved with a rising or falling signal edge even if t1=t2, and therefore do **not** constitute a conflict.

The example network can now be revisited and analyzed for dynamic signal propagation. The focus is on the highlighted path through the network that static sensitization claimed to be a false path.

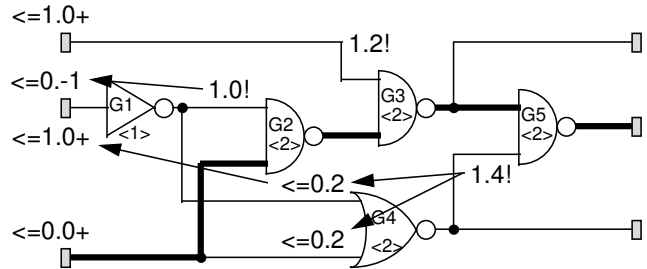


Figure 5: Dynamic Path Sensitization Example

Figure 5 shows the results of some value implications when trying to propagate a signal edge along the highlighted path. The crucial implications are indicated by the arrows in the figure. <=0.-1 is needed at the second network input to sensitize the path through G2. The path is easily extended through G3 by the <=1.0+ on the topmost network input (since the network input must be stable after time 0, a <=1.0+ is needed to satisfy the required 1.2! sensitization value at G3). Further sensitization through G5 requires a 1.4! on the lower input of G5, implying <=0.2 on both inputs of the NOR gate G4. These, in turn, can only be satisfied with a <=1.0+ from the second network input and a <=0.0+ from the third network input. The second network input now has two value requests, <=0.-1 and <=1.0+, which cannot be resolved statically, but can be resolved in time. A rising transition at time 0, for example, will do the trick. The 0.0+ on the third network input indicates that the critical signal edge that gets propagated is a falling edge.

An example of a path that fails dynamic sensitization is illustrated in Figure 6.

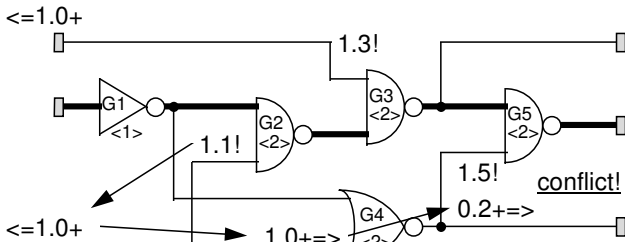


Figure 6: False Dynamic Path

The 1.1! on the lower input of G2 implies a $\leq 1.0+$ for the third network input (the signal must be stable after time 0). This value forces the output of the NOR gate G4 to $0.2+=$, blocking the continuation of the path through G5 for all times later than time 2. Hence, the highlighted path cannot be sensitized through G5. The unresolvable conflict is a consequence of the one-sided stability assumption at the network inputs.

To summarize, the dynamic sensitization procedure identifies which paths can propagate signal edges. It also can be used to derive test wave forms, if the path can be sensitized. The longest dynamically sensitizable path in the example network is found to be 6 time units long. That is shorter than the value predicted by static timing analysis (7 time units), but longer than what static sensitization analysis suggests (5 time units).

4. Timing Sensitivity and Slack

It has been mentioned earlier that slack values probably are the most useful output from timing analysis. The path sensitization analysis confirms that the slack values calculated by static timing analysis as shown in Figure 1 are pessimistic. The longest path in the network (7 time units) can neither be statically nor dynamically sensitized. Hence, the signal at the second network output stabilizes after time 6, and no timing violation occurs for the given required arrival time.

How can the Slack values be adjusted to reflect this?

The authors are not aware of any publications addressing the problem of how slack can be re-calculated after false path identification and removal. Hence, the following discussions begin to enter uncharted waters.

In static timing analysis, slack is calculated from the predicted local signal arrival times and the required local arrival times as was illustrated in Figure 1. The main problem is that after false path identification the local required and predicted arrival times are no longer valid. For example, pushing the required arrival times back from the second network output along the highlighted path in Figure 1 is meaningless, because no

signal edges can be sensitized along that path. Unfortunately there seems to be no easy way to reconcile the desire for a simple and efficient local algorithm with the need to include path-specific information -- and, unfortunately, this paper does not suggest one either. Instead, the following discussions concentrate on illustrating some non-obvious results derived from dynamic path analysis, which need to be considered when trying to re-calculate slack values.

The slack for a node in the network indicates how much delay can be added (positive slack) or must be subtracted (negative slack) to exactly meet the required signal arrival time. That is, it should be possible to calculate the slack value for a node by adding an incremental time value ϵ to the local AT and see how this affects the output AT. The output AT now becomes a function of ϵ which represents the sensitivity of the output AT to local AT changes at the source node. The simple additive arithmetic used in static timing analysis results in linear sensitivity functions (with respect to ϵ). This is illustrated in Figure 7:

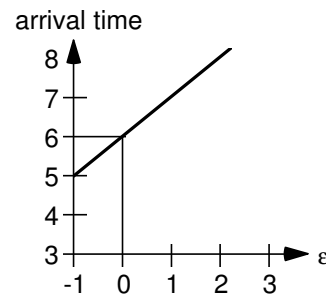
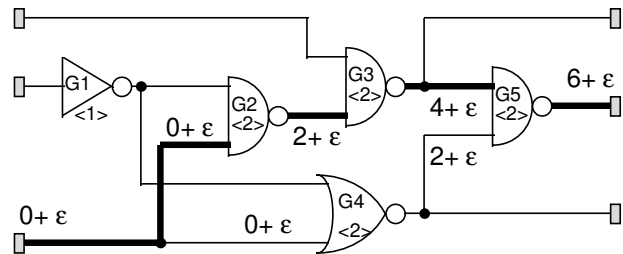


Figure 7: Output Sensitivity Function Example for Static Timing Analysis

The sensitivity graph depicts how the second network output depends on the bottom network input AT. The dashed lines illustrate how the sensitivity function relates to slack. The ϵ -value corresponding to the RAT is equivalent to the Slack for this RAT. With a RAT of 6, for example, the slack value for the bottom network input is 0, consistent with Figure 1.

Besides linearity, the sensitivity functions corresponding to static timing analysis have another nice property: the sensitivity function for a network input is totally independent of signal arrival times at other network inputs. That is, adjusting the arrival time for one network input does not require re-calculating the sensitivity functions for the other inputs. When dynamic sensitization effects are taken into account, the picture changes dramatically. It already has been shown that signal edge propagation can very well depend on the switching of other signals. The interesting question is how that affects the properties of the sensitivity functions.

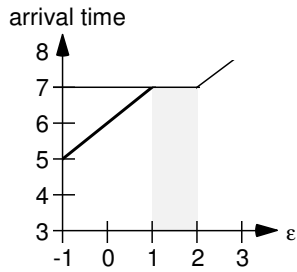
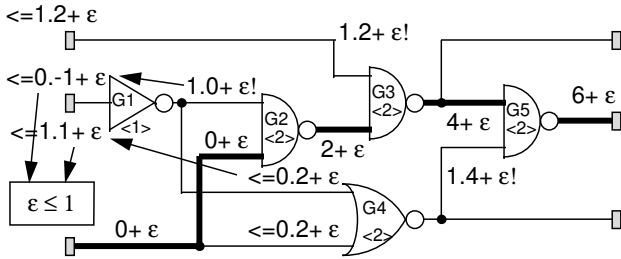


Figure 8: Dynamic Sensitivity Function Example

Figure 8 illustrates the output timing sensitivity with respect to dynamic path sensitization. Sensitizing the highlighted path from the lower network input to the second network output implies $\leq 0.-1+\epsilon$ and $\leq 1.1+\epsilon$ at the second network input. Since the network input is not assumed to switch after time 0, it is only possible to resolve the two value requests if $\epsilon \leq 1$. When ϵ gets larger than 1, the output timing is controlled by the second network input and stays at 7 time units, until for ϵ larger than 2 the third network input becomes dominant again via the alternate path through G4 and G5. The resulting output sensitivity function no longer is linear, and the slack for a RAT of 7 becomes ambiguous. Judging by AT values alone, the slack should be 1 (the latest dynamically viable signal activity at the second network output occurs at time 6). The dynamic analysis, however, indicates that the third network input could be delayed by as much as 2 time units and still meet a RAT of 7 time units.

Similar calculations can be made for the other network inputs and outputs. The second network input, for example, has an apparent slack of 2 relative to the second network output (the path along G1-G2-G3-G5 cannot be sensitized, leaving the 5 time units long path through G1-G4-G5), and a slack of 1 relative to the first network output. Hence one might be tempted to conclude that the second network input can be delayed by 1 time unit (constrained by the first network output), and the third output can be delayed by 2 time units based on the graph shown in Figure 8. That, however, is wrong. The sensitivity graph shown in Figure 8 is only valid if the second network does not switch after time 0. Hence, the sensitivity function for the third network input would have to be re-calculated based on the changed timing at the second network input (and vice versa).

5. An Extreme Example

In extreme cases, the output sensitivity function and the associated slack can become even more unusual. The example in Figure 9 illustrates some pathological behavior.

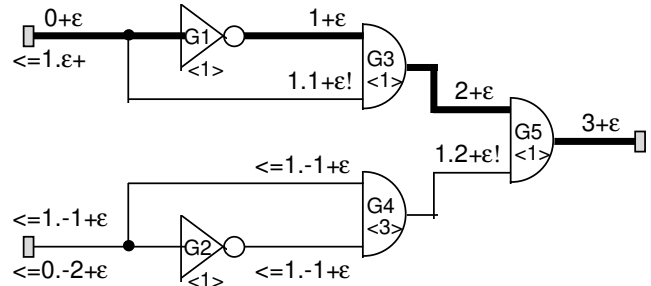


Figure 9: Pathological Example

The input transition at the lower network input for this example is assumed to occur exactly at time 0. Figure 9 shows the implications for deriving the output sensitivity function relative to the upper network input. The implications show how a rising edge at time ϵ can be propagated along the highlighted path. To propagate this edge, the lower network input must meet $\leq 0.-2+\epsilon$ and $\leq 1.-1+\epsilon$. With the more stringent requirement that the lower input must switch exactly at time 0, the two constraints for the lower network input can only be resolved if $1 \leq \epsilon \leq 2$.

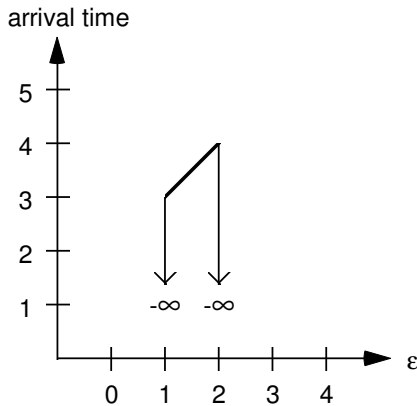


Figure 10: Sensitivity Function for Example from Figure 8

If ϵ is smaller than 1 or larger than 2, then no edge can propagate to the output. This raises the question of what is the arrival time in this case? The most natural choice is to use $-\infty$ as arrival time for non-switching signals. Figure 10 shows the output sensitivity function corresponding to the example in Figure 9.

6. Dynamic Path Sensitization and Small Delay Fault Testing

The sometimes bizarre non-linear behavior of the output sensitivity functions depicted in the previous chapters is closely related to similar effects in small delay fault testing. So called block-oriented small delay faults [6] can be modeled as incremental adders to the local signal arrival times, very much like the incremental delay adder ϵ used for sensitivity analysis. The main difference between the analysis presented here and the one used in small delay fault simulation is in deciding what constitutes a decisive event. A small delay fault is unambiguously being detected only if the added delay forces a stable wrong value on the net at the time of measurement (the fault, so to speak, must push the trailing edge of a signal transition region over the measurement time). In addition, the activation of a delay fault requires (by definition) a signal transition not just a glitch. Timing sensitivity analysis, by contrast, is concerned with potential timing failures not only guaranteed failures. Any signal instability (from the leading edge of an instability region on, regardless of whether it is activated by a transition or a glitch) can cause a potential timing failure. In other words, although the concept of delay adders is very similar, the criteria for "failure" are much less stringent in timing sensitivity analysis than in small delay fault simulation.

7. Conclusions

It has been reviewed how the use of dynamic effects must be taken into account for the identification of false timing paths. The concept of timing sensitivity was introduced to illustrate how the consideration of dynamic effects can lead to non-linearities in and interdependencies between signal Slack contributions. The findings suggest that great care must be applied when trying to re-calculate Slack values after false paths have been identified and removed.

8. Acknowledgments

The work reported in this paper was performed while B. Könemann was with IBM, Corp., in Poughkeepsie, NY.

9. References

- [1] J. Benkoski, E. Vanden Meersch, L. Claesen, and H. DeMan, "Efficient Algorithms for Solving the False Path Problem in Timing Verification", Proceedings ICCAD, pp. 44-47, 1987.
- [2] R. Hitchcock, G.L. Smith, and D.D. Cheng, "Timing Analysis of Computer Hardware", IBM Journal of Research & Development, Vol. 26, No. 1, pp.100-105, January 1982.
- [3] D. Brand, and V.S. Iyengar, "Timing Analysis Using Functional Analysis", IEEE Trans. Comput., vol. 37, pp. 1309-1314, 1988.
- [4] McGreer, and R.K. Brayton, "Efficient Algorithms for Computing the Longest Viable Path in a Combinational Network", Proceedings 26th Design Automation Conference, pp. 561-567, 1989.
- [5] H.Chang, J.A. Abraham, "VIPER: An Efficient Vigorously Sensitizable PathExtractor", Proceedings 30th Design Automation Conference, pp. 112-117,1993.
- [6] V.S. Iyengar, B.K. Rosen, and J.A. Waicukauski, "On Computing the Sizes of Detected Delay Faults", IEEE Trans. Comput. Aided Design, col. 9, No. 3, 1990.