A Three-tier Assertion Technique for Spice Verification of Transistor Level Timing Analysis

Savithri S., *savithri@miel.mot.com*
Motorola India Electronics Ltd., Bangalore, India

David G. Blaauw, *blaauw@adttx.sps.mot.com*
Abhijit Dharchoudhury, *abhijit@adttx.sps.mot.com*
Advanced Systems Technology Lab, Austin, Texas (US)

Abstract

Static transistor level timing analysis has become more and more accepted method for performance evaluation because of its reduced design cycle time when compared to the vector based timing analysis. These static timing analysis tools use transistor level delay modelling to identify timing-critical paths and estimate performance of the design. With increase in complexity of designs it becomes necessary to verify the timing-critical paths using spice-simulations. In order to perform accurate modelling for spice simulations, it becomes imperative to identify all the devices and the signal-states on the nodes, for a given input to output transition. The current techniques use greedy approaches for each input to output transition in a channel connected component. These techniques consider turning-on all transistors on the primary conducting path and turning-off remaining transistors on the side-paths. Thus, these techniques don't consider appropriate loading on the output node due to transistors on the side-paths and the fanout paths. These techniques also don't consider the input signal correlations. This paper presents a three-tier heuristics to determine side path assertions, such that it maximizes, as much as possible, the load at the output node, for a given set of input to output transition. Also, a method to perform the fanout path assertions is presented. This technique has been used for spice-verification of the timing-critical paths of transistor level designs. The results have been compared using spice simulations of the same designs.

Keywords: *assertion, assertibility, spice verification, timing analysis, primary-path, secondary-path*

1. Introduction

With increased complexity and time-to-market pressures, static timing analysis is becoming critical in the verification of high performance VLSI designs. Static timing analysis implicitly verifies all paths in a design and is therefore more reliable than dynamic timing analysis where the degree of verification coverage depends on the completeness of the simulation vectors.

Many synthesis and optimization tools use static analysis as its backbone. Static timing analysis assumes the design is functionally verified and identifies performance bottlenecks in the design. It is used to eliminate timing errors, determine timing-critical paths and analyze design performance.

Transistor level timing analysis involves, dividing the design into DCC[1]s and then, analysing each DCC for worst-case delays for given input-output transitions. This is achieved by tracing the paths from output node to rail node (vdd/gnd) for the DCC. The current techniques of path tracing use greedy approaches where all the transistors on the conducting path are turned-on and all remaining transistors in the side paths are turned-off. This method may under-estimate the delay since additional loading provided by the transistors in the side-path may be ignored. Also, these techniques need to consider the input signal correlations, which might arise due to logical dependency of the signals (see [1]). Evaluation of signal-correlation of all the inputs for a DCC is a very difficult problem by itself. Consideration of signals along the path becomes more difficult. Hence an optimal method to choose the set of input signals need to be devised. In this paper a three-tier heuristic is presented for turning-on/off the transistors on the primary, secondary and fanout paths at transistor level delay modelling. The earlier techniques used to set the inputs for each DCC in turn,whereas our approach takes the whole path into consideration for setting the inputs. This allows a more accurate modelling of the signal correlations along the timing critical path. The three tier technique presented in this paper, has been used in spice-verification of timing critical paths for the state-of-the-art industrial designs. The organization of the paper is as follows : some preliminaries and problem definition are developed in the following section and in section 3, the new technique is described. Sections 4 and 5 illustrate the spice verification results and conclu-

1. a DCC (DC connected component) is a set of source-drain (channel) connected transistors without considering connections through vdd/gnd

sions, respectively.

2. Preliminaries

The delay calculation at transistor level analysis involves partitioning the design into directly connected components (DCCs). The DCCs are topologically sorted. Each DCC is then considered for delay-calculation using path-tracing and delay-modelling techniques. The timing-critical paths are then identified, which is fixed by the designers (either using sizing/restructuring tools or manually fixing the transistors on the critical path) for improving performance. However, the critical-path that is reported by the delay-calculator can introduce some path errors (see [1]). To verify these timing-paths for correctness manually can result in wastage of design resources. Hence, an automatic spice-verification of these paths need to be implemented. In order to accurately model for spice verification, it is required that all the transistors and the nodes that contribute to the critical-path are included for spice simulations. And, also, the input signal-states for the transistors of all DCCs along the timing-path have to be determined. This involves determination of appropriate initial conditions for the circuit. For complex logic structures, it becomes imperative to include the logical dependencies across DCCs while generating the spice-verification deck. The problem is formulated in the following paragraphs of this section.

Definitions

1. *A Trigger* is defined as the transistor whose gate-node is switching, which causes an output transition; e.g. in figure 1, transistor **n1** is switching from low to high values.
2. A P*rimary path* is defined as the path from rail node (vdd/gnd) to output node that contains the trigger (e.g., path through the transistors **n1** and **n2**).
3. A S*econdary path* is defined as a non-conducting path from rail node to output node, that may or may not include the trigger. This includes all side-paths from rail (vdd/gnd) to output nodes, excluding primary path.

Consider an example in figure 1, a single DCC of an XOR circuit. This DCC has a set of inputs (**a1** and **a2**) and a set of outputs (**out**). Given that transistor **n1** is trigger (that is, gate-node **a2** transitions from low to high) - for the output **out**, to switch from high to low, there are several paths from the rail (vdd/gnd) to the output node (through, **n2** and **p2** transistors of the inverter). The requirement here is to determine valid set of input-signals, for the transistors to be turned-on/off in the DCC for given input-output transition to happen. In other words, we need to determine the valid signal-states for nodes on the primary and secondary paths of the DCC. For example, for the transistor **n1** switching from low to high value and output **out** falling from high to low , it is required to find the signal-states on the nodes **a1** and **int**, which determine the set of transistors that are turned-on as well as those transistors that are turned-off. Along with this, it is required to find the signal-states of all the nodes in the fanout DCCs of this XOR-DCC.

Thus, the aim is to identify an optimal-delay (maximal/minimal-delay depending on worst-case or best-case analysis[1]) primary-path for given input-output signal transition. And, determine an optimal set of signal-states for the nodes on the secondary and fanout paths of the DCC. To determine such an optimal set of signal-states on all the nodes of the DCC is as hard a problem as the dynamic sensitization problem, that is, NP-hard [1]. This is because of the input signal-correlations , including the logical dependencies and the reconvergent nodes along the timing path.

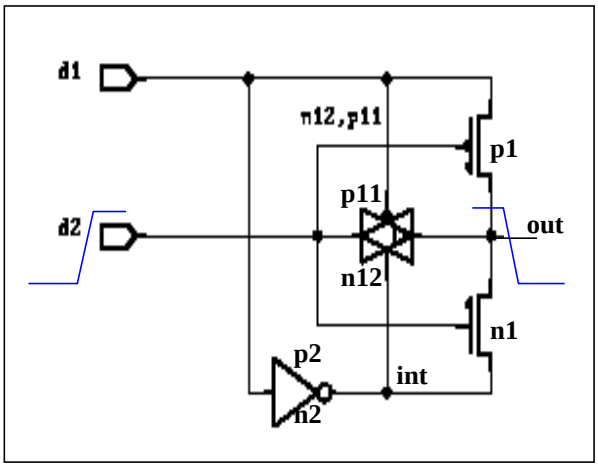

Figure 1 XOR: a single DCC

Problem statement

For a given timing critical path, the problem is to determine the signal-states of the nodes on the primary and secondary paths of all DCCs on the timing-path. Also, determine the set of valid signal-states for all fanout DCCs. These are determined such that following conditions are handled:

• C1 • More than one path from rail to output node through the trigger should be resolved

• C2 • Secondary paths should be non-conducting and should consider accurate loading conditions

1. all the analysis discussed further are only on worst-case analysis; it can be easily extended for the best-case analysis as well.

• C3 • Fanout paths should be handled appropriately to include the miller capacitances

• C4 • Signal correlations (for mutually exclusive signals, inversions etc.) need to be handled

The technique described below attempts at resolving these conditions based on a heuristic, such that there is as much maximization of the load as possible at the output-node.

3. Three-tier Assertion technique

Definitions

1. A*ssertion* is defined as assignment of one of the following signal-states to a node, as LO, HI, Lo2Hi or Hi2Lo.
2. *Assertibility* is defined as the feasibility of an assertion on the node.

The signal-states that are defined for a node are LO (signal low), HI(signal high), Lo2Hi (rising signal) and Hi2Lo (falling signal). The default state is Undef (undefined). During path-tracing a node might be asserted to HI state; the same node could be re-visited with a new state, say, LO. This results in a conflict in the signal-states as the same node cannot be signals low and high simultaneously; and the node is said to be non-assertible. However, if the same node is revisited with Lo2Hi state, then, the node is assertible, since the final state is still signal-state high. Every time a node is assigned a signal-state (i.e., node is said to be asserted), its assertibility is verified before assignment.

With this understanding of assertion and assertibility, the problem stated in the previous section can now be translated to determine the following parameters for each DCC :

• P1 • input assertion for the transistors (and, hence, all nodes) on the primary path

• P2 • input assertion for the transistors (and, hence, all nodes) on the secondary path

• P3 • fanout-node[1] assertions at all the output-nodes of the DCC

• P4 • initial conditions on all the switching nodes

The three-tiers of the heuristics are described in sections 3.1, 3.2 and 3.3. These steps determine the parameters P1, P2 and P3 respectively. To determine the last parameter, P4, all the node assertions are first obtained and then, depending on the signal-transition (Lo2Hi or Hi2Lo) of the nodes, the initial conditions for the nodes are assigned (as LO for Lo2Hi and HI for Hi2Lo). The procedure is described briefly below. The three-tier algorithm is illustrated in figure 3.1.

For the given timing critical path, topologically sort the DCCs from the timing input to the timing output. Then, for each DCC perform the following operations. This task of assertions is divided into three basic operations (and, hence, termed as three-tier technique).

1. Determine the primary paths for all the DCCs on the timing-path under consideration and update the assertions on these primary paths.
2. Identify the set of transistors that can be asserted such that there is maximum loading at the output node, at the same time, there are no conducting paths to the rail (excluding the sneak paths and the split primary paths[2]) for all the DCCs in the path.
3. Determine all the fanout DCCs and perform the primary-path and secondary path assertions.

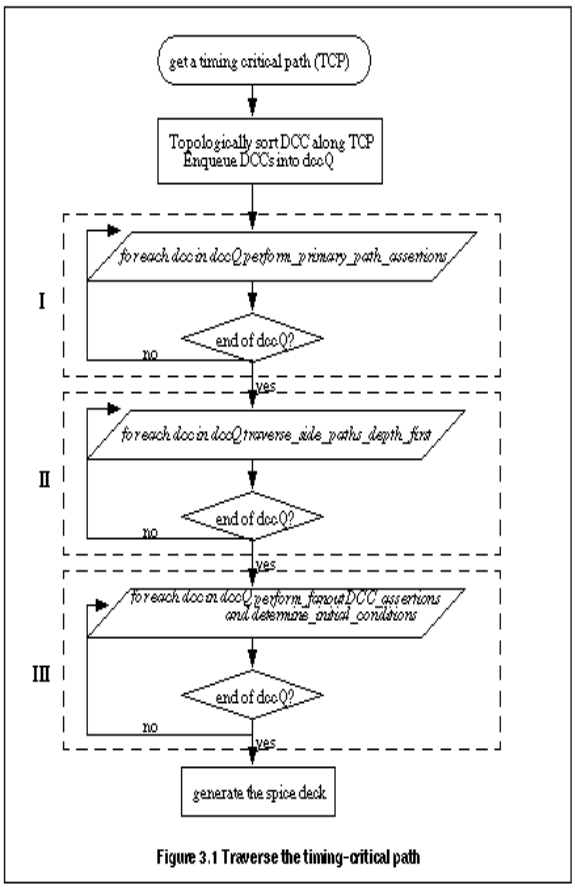

Figure 3.1 Traverse the timing-critical path

1. All nodes in the fanout DCCs are collectively termed as fanout-node.

2. described in section 3.1

It is important to note that these operations are performed in that order. That is, for a given timing-critical-path, the assertions on the primary paths for all DCCs are performed first; then, the assertions on the secondary paths for all the DCCs are performed. Finally, the assertions on the fanout DCC paths are performed. After all these three operations are completed, the initial conditions are determined for all switching nodes as described previously. A spice deck is generated for all the DCCs (both, along the timing critical-path and all the fanout DCCs that were considered in the third-tier above). Each of these operations are elaborated in the following sections.

3.1. Primary Path assertions

Given a trigger transistor, there can be more than one path through the trigger that connects the output node to rail. We select the most optimal path by choosing the worst-delay path through the trigger. The directions of all the transistors are determined while tracing the path. All the transistors along this path are turned-on. All the source-drain nodes from the rail, till the trigger transistor is found are steady states (LO or HI). The remaining nodes, from the trigger to the output-node are switching (Hi2Lo or Lo2Hi, depending on if the path is electrically connected to gnd or vdd, respectively). If there are more than one path through the trigger, then, that path with worst delay is chosen. The algorithm is illustrated in figure 3.4.

While performing the assertions all the signal correlations are taken into consideration. Note that, this might result in a path that is conducting. For example, a balanced-delay structure as in figure 3.2, will enable all the three parallel paths. We classify these parallel-paths for assertions as *split-primary-paths*.

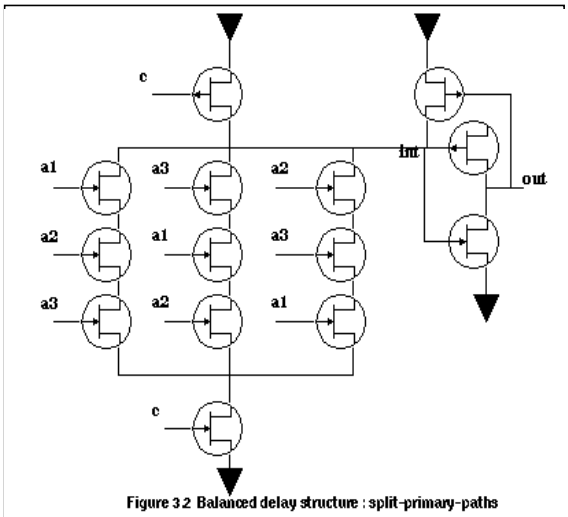

Figure 3.2 Balanced delay structure : split-primary-paths

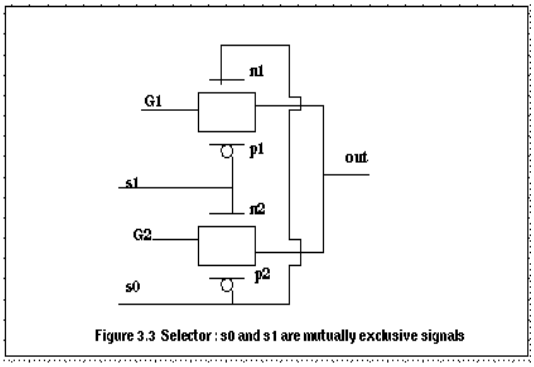

Figure 3.3 Selector : s0 and s1 are mutually exclusive signals

The assumption made here is that all the paths are assertible. But, due to certain signal correlations, there might be certain paths that become insensitizable. For example, in figure 3.3, the signal **S1** is an inversion of signal **S0**.

```
Initialize max_delay = 0.0; trigger_found = false
For each path towards gnd (vdd)
   For each curr_transistor in path
      if (trigger_found)
        assert the drain-node of curr_transistor to LO (HI)
       else
        assert the drain-node of curr_transistor to Hi2Lo (Lo2Hi)
       endif
      if ( curr_transistor is trigger)
         trigger_found = true
        assert gate-node of curr_transistor Lo2Hi (Hi2Lo)
       else
        assert gate-node of curr_transistor HI
       endif
   End curr_transistor
   find delay of path, set in path_delay
   if (path_delay > max_delay)
      store max_path = path
      set max_delay = path_delay
End path
```

Figure 3.4 *perform_primary_path_assertions*

3.2. Secondary Path assertions

Once the primary path has been identified, all the side paths from rail node to the output node other than the primary path are identified. The nodes on these paths are asserted to values such that the paths are non-conducting. The conventional methods turn-off all the transistors on these path. This can result in a grossly pessimistic approach. We present here some heuristics for assertions on the secondary paths to overcome this.

The idea behind the heuristic is to load the output-node

as maximum as possible. For this, turn-off only one transistor along the path that is closest to the rail-node. Such that, there are as many transistors conducting as possible that are source/drain connected to the output-node. This is achieved by traversing all the secondary paths in a depth first manner. The algorithm is illustrated as traverse_side_paths_depth_first (a recursive algorithm) in figure 3.5. This algorithm also takes care of split-primary paths as conducting paths and not as secondary non-conducting paths.

```
Initialize atleast_one_path_to_rail to false

For each curr_transistor that is drain-connected to output-node

   found_path_to_rail = traverse_side_paths_depth_first

   if ( found_path_to_rail is true )
      try to turn curr_transistor off
   else
      try to turn curr_transistor on
   endif

   if (curr_transistor is on and found_path_to_rail is true)
     set atleast_one_path_to_rail to true
   endif

End curr_transistor

return( atleast_one_path_to_rail )
```

Figure 3.5. *traverse_side_paths_depth_first*

3.3. Fanout assertions

This step involves performing fanout DCC assertions. In this, all the fanout DCCs are identified and for each of those DCCs steps 3.1 and 3.2 are carried out. The fanout DCCs identified are only one level deep from the current DCC. These assertions satisfy condition C3. The algorithm is illustrated as perform_fanoutDCC_assertions in figure 3.6.

```
For each DCC that is fanout to the output-node

      perform_primary_path_assertions
      traverse_side_paths_depth_first
End DCC
```

Figure 3.6 *perform_fanoutDCC_assertions*

The three-tier assertions described above allows maximizing(for worst-case analysis) the load at the output node. However, the path selection for the depth first traversal in secondary path assertions is done arbitrarily. That is, there might be a different set of path selection that can result in a more optimal loading conditions at the output node. The signal correlations can be either derived automatically (using BDD representations etc.) or user-defined. Effective assertion as a result of the above operations is stored in each node. If assertions conflict on the primary path then the timing critical path becomes insensitizable. Assignment of transistor's directions will also affect the determination of the paths and assertions [2].

4. Results

The technique described in the previous section was carried out using an existing in-house static timing analysis tool. Also, the spice simulations were carried out using an in-house spice tool. The spice deck was generated for various circuits and verified using spice simulations. The mutually exclusive relations on the signal inputs were provided by the user. The experiments were carried out on a Solaris2.5 platform on large set of industrial designs. The comparison table for few example designs is given below in Table 1. The table shows deviations for the topmost timing-critical path. Experiments were carried out on different timing critical paths for large set of designs. The results indicate that the deviations for most of the circuits were - less than 2% for most designs; and, less than 5% on an average of these designs.

Table 1: Comparison of spice-verification with spice results for top timig-critical path

Design	Spice-based critical-path delay	Spice Verification for the critical-path	%Deviation
xor	1.302ns	1.298ns	0.25%
xor2	1.647ns	1.619ns	1.74%
conditional mux	5.819ns	5.308ns	9.63%
adder	1.001ns	0.997ns	0.36%
and-or-invert	1.740ns	1.731ns	0.56%

5. Conclusions

With the increasing complexity of designs and existence of pessimistic delay models, it is necessary that the critical paths reported by timing analysis are verified with spice simulations. Spice simulations of complex and large de-

signs become very expensive; hence, a method to verify the timing critical paths becomes necessary. In order to accurately generate spice deck for simulations, it is necessary to incorporate all the signal dependencies. The results are directly related to the specific input signals used to drive the simulator. Hence, the deck should consider accurate initial conditions with the signal correlations and include all dependent transistors and nodes for the delay computation. The assertion technique described in this paper models this accurately for spice verification. The following are some inferences from the three-tier assertions:

- All transistors on the primary path are turned-on. This allows a path from rail to the output-node. If there are more than one path which include the trigger transistor, then the path with maximum[1] delay is considered.
- The nodes on the secondary paths are asserted such that there is maximum possible loading at the output node ensuring that the path is non-conducting (excluding, sneak-paths and split-primary-paths).
- The fanout DCCs are asserted such that the miller capacitances are maximized.This is achieved by performing the primary and secondary path assertions on the fanout DCCs.
- All nodes on the primary paths of all the DCCs for a given timing critical path are asserted before secondary path assertions to avoid any non-assertibility of the DCC output-nodes.
- All fanout assertions are carried out after primary and secondary assertions for the given timing critical path.

The three-tier assertion technique presented here is quite generic. And, the technique can also, easily, incorporate any signal correlations. With this technique now verified using spice simulations, it can be easily extended for any transistor level delay modelling. This technique can be extended for best-case analyses as well.

Acknowledgements

The authors wish to thank the Focus team at Bangalore and Austin for their contributions to the success of this project.

1.all discussions in this paper are for the worst-case analysis. For the best-case analysis, the methods remain the same; but, the assertions should consider minimum delay-path and minimum capacitive loading.

References

1. Desai, M.P., and Y.T. Yen, " A systematic technique for verifying critical path delays in a 300MHz Alpha CPU design using circuit simulation", *Proceedings of the Design Automation Conference*, 1996, pp 125 - 130.

2. Lee, K.J., C.N.Wang, R. Gupta, and M.A. Breuer, " An integrated system for assigning signal flow directions to CMOS transistors", *IEEE transactions on Computer Aided Design*, **vol. 14**, No. 12, Dec.'95, pp 1445-1458.

3. Elmore, W.C., " The transient response of damped linear networks with particular regard to wideband amplifiers", *Journal of Applied physics*, **vol. 19,** Jan.'48, pp. 55 - 63

4. J.J. Grodstein, J.Pan, W. Grundman, B. Gieseke, and Y.T.Yen, "Constraint Identification for Timing Verification", *Proceedings of International Conference on Computer-Aided Design*, pp. 16-19, Nov. '90.

5. T. Burks and R.E.Morris, "Incorporating Signal Dependencies into static transistor-level Delay Calculation", *Proceedings of International Conference on Circuit Design*, 1996.