Tau White Paper

Board-level circuits today routinely run at speeds of 100 MHz or more and are composed of dozens of complex interacting VLSI components. To design such circuits in a timely and correct manner it is necessary to pay close attention to circuit timing early in the design cycle. At fast clock speeds, managing component and interconnect propagation delay becomes a key aspect of circuit design. It is imperative that the critical paths on a circuit and the slack available for interconnect delay consumption be identified early, and drive subsequent stages in the design flow.

Simulation as an approach to analyzing circuit timing is cumbersome, time-consuming and resource-intensive. Besides, the extent to which timing problems are discovered using simulation is entirely dependent on the stimuli provided by a design engineer, i.e. if the stimulus that causes a timing problem to manifest itself is not simulated, then the problem will remain undetected. To obtain complete results in an efficient manner, design engineers have increasingly turned toward verification technology. Verification approaches accept as input a circuit description, requirements on circuit behavior, and automatically (without user-supplied stimulus) determine if the requirements are satisfied. Since design engineers do not provide any stimulus, they do not impact the completeness of verification results. Instead, verification accuracy is dependent on the proximity between models of component behavior and the behavior actually exhibited by a component.

Static timing analysis is the most commonly available commercial solution to the timing verification problem. Static analysis was developed to address the gate-level timing verification problem, where a circuit is composed of combinational logic blocks (CLBs), flip-flops and latches, with the objective of verifying the satisfaction of set-up and hold times. When modeling component behavior, static analysis ignores functional information (e.g., the Boolean function computed by a CLB output), while capturing temporal information (timing delays from input to output, set-up/hold times, etc.).

By ignoring functional information, static analysis is able to efficiently analyze the timing characteristics of realistic gate-level circuits. However, for the same reason, static analysis reports timing violations, called false violations, that represent fictitious problems. False violations result from traversing false paths during static analysis. A false path cannot have a transition propagate through it given the functional behavior of components along that path. False violations are problematic because they require design engineers to do one of two things:

1. Manually sift through the list of violations reported by a static timing analyzer and determine which violations are false. This task is cumbersome and potentially error-prone.

2. Assume that most violations reported by a static timing analyzer are true. This results in conservative designs that do not push performance limits, and wasted effort in unnecessarily changing error-free portions of a design.

Extending static analysis to automatically eliminate false violations is computationally prohibitive at the gate level of granularity. Instead, it is expedient to tolerate these violations and use designer insight to work around them. At the board level, however, it's a different story. Unlike gate-level circuits that have thousands of simple components, board-level circuits contain dozens of complex components, such as processors, memory controllers, and ASICs. These components exhibit sophisticated interface behavior, performing read, write, and bus-arbitration cycles. Ignoring the functional information of such behavior (for example, when ready is asserted, a read cycle is terminated and data is latched) causes meaningless results: most violations that are reported tend to be false, and more seriously, the true behavior of a circuit is not analyzed.

To obtain more realistic results at the board level, static analysis has been extended to perform case analysis. With this approach, the interface behavior of a complex component is described in terms of cases, such as read and write cycles, and design engineers select the appropriate cases under which a circuit is analyzed (for example, processor and memory performing read cycles). Unfortunately, case analysis is more simulation-like than verification. By selecting the cases under which a circuit is analyzed, design engineers directly influence the completeness of verification results. After all, the toughest problems to detect are those that occur as a result of unanticipated, or corner, cases.

So, static analysis and its extensions are unsuitable for board-level timing verification. To address this situation, Interconnectix has developed an accurate and efficient board-level timing-verification tool, Tau(TM), based on a new methodology called *symbolic timing analysis*.

Symbolic analysis takes as input a circuit description, and for each component in the circuit a bus-functional timing model. In addition to capturing timing delay and constraint information, bus-functional timing models encapsulate functional aspects of component interface behavior. For example, they capture the manner in which control and data signals switch over time during read and write cycles, or how, during a read cycle, it is necessary to satisfy set-up and hold times on data only when ready is asserted. In effect, bus-functional timing models provide a realistic description of component interface behavior for the purpose of timing analysis.

Given a circuit description and bus-functional timing models, symbolic analysis automatically enumerates the different temporal behaviors on a circuit, and the conditions under which these behaviors occur. Timing constraints that apply to each temporal behavior are verified. The conditions are used to maintain consistency during analysis, i.e. to prevent analyzing the impact of two behaviors that are mutually inconsistent. The conditions are also useful when reporting timing violations, because they define the stimulus required to cause a violation.

Symbolic analysis reports accurate results because it uses bus-functional timing models, rather than just timing delay and constraint information. When bus-functional models accurately capture component interface behavior, symbolic analysis identifies all timing violations on a circuit without reporting any false violations. Also, symbolic analysis accepts and verifies asynchronous behavior and constraints. Circuits are allowed to have fully asynchronous clocks, and asynchronous timing constraints on components are verified.

Symbolic analysis is efficient because it separately analyzes only those behaviors that are temporally unique. Temporally equivalent behaviors are grouped together. In this manner, symbolic analysis implicitly enumerates the space of possible circuit behaviors, i.e., exhaustive results are provided without separately, or explicitly, examining all individual circuit behaviors. Bus-functional timing models also facilitate efficient timing verification because they abstract implementation details, thereby reducing the level of granularity at which analysis is performed.

When compared to static analysis as applied to board-level circuits, symbolic analysis offers significantly more accurate results with comparable efficiency.

In addition to the benefits resulting from the use of symbolic timing analysis, Tau offers design engineers another key advantage. Tau is closely integrated with the Interconnect Synthesis (IS) tool suite provided by Interconnectix. Together, Tau and IS provide design engineers a complete timing-driven physical design solution.

Design engineers use Tau to automatically identify the critical paths on a circuit and the slack available for interconnect delay consumption. This information is obtained well before physical design commences. As Tau takes functional information into account, false critical paths are not reported, thereby allowing design engineers to focus on the real physical design challenges posed by their circuits. Information on the critical interconnect paths of a circuit are exported from Tau to IS. Then, physical design is performed within IS, where the interconnect delay constraints are used to guide design engineers toward a correct solution. It is possible to back-annotate interconnect delay values from IS to Tau, allowing timing analysis to be performed taking actual interconnect delay into account. Tau's integration with IS significantly reduces the number of iterations required to obtain a physical design that is consistent with the timing requirements of a circuit.

To summarize, Tau offers the following value to design engineers:

> **1.** A reduction in both the time and resources required to perform circuit design by using a timing-verification tool that does not report false violations.
>
> **2.** By driving the physical-design process, the iterations required to achieve a circuit implementation that meets timing requirements are significantly reduced.
>
> **3.** The ability to push the performance limits of designs by using a timing verification solution that provides accurate results.

For information on Tau modeling, go to the Tau Modeling page.