

Timing closure remains one of the top issues facing design teams today.

Consequently, it is no surprise that the timing simulation of gate-level designs is still a common practice in IC verification.

If gate-level simulations are run, they often reveal timing-related bugs that need to be debugged, analysed, and fixed by design teams.

However, practical debug requires recording an enormous amount of signal data. Despite the tremendous computing horsepower available, gate-level timing simulation is notoriously slow as it is, and capturing enough data for debug further exacerbates the problem. Indeed, the problem of visibility we talk about with verification at the register transfer level is at least as bad or more likely worse at the gate level. Gate-level net lists typically have many more signals, and these signals have much more activity (events) to capture for debug and analysis.

The visibility problem with gate-level netlists has been around for a while and as such, design and verification teams have developed methodologies to address this. At best, these can be classified as heuristics-driven workarounds. Some of the workarounds are ad-hoc while others are more systematically developed, but none address the problem fully and optimally while maintaining the desired end-result of full visibility. The motivation for chip designers and EDA vendors to overcome this obstacle is significant. The current workarounds are painful but necessary. Some of these workarounds are based on limiting gate-level timing simulations to only the most critical tests and paying the price of long, slow re-simulations to capture data when problems are detected. Others try and guess, sometimes somewhat intelligently, the scopes and/or time windows to record data thereby reducing the simulation overhead but at the price of multiple simulation iterations to capture the next scope and/or time-window of interest. Such workarounds provide short-term fixes, but ultimately do not scale with the increasing complexity and size of SoCs.

The systematic reduction of simulation overhead by utilising visibility enhancement techniques is now well-accepted and proven. For example, the Novas Siloti technology from SpringSoft has shown significant gains in verification throughput for many large designs. The technique employs two engines that work hand-in-hand. The core engine is the Data Expansion engine that can compute values of a majority of the

design nodes as long as it has access to a smaller set of "essential" signals. These essential signals are algorithmically determined prior to simulation and typically include all the sequential register outputs, primary I/O, and other values that cannot be easily derived by the data expansion engine. The simulator is directed to only record these nodes resulting in file and time savings in the order of 4x. The highly optimised dump file is then put into a debug system that then queries the Data Expansion engine for values that have not been recorded during simulation. The query and

window (and optionally the scope) for further analysis requiring a timing-accurate record. By definition, the initial pass based on a data expansion engine that ignores timing can be used to accurately trace a problem down to a particular logic cone – for instance, a cone whose output exhibits an error while none of its inputs do. In such situations, the culprit is often some timing issue within the cone. Alternative means of narrowing down the time-window of interest can also be employed – for example, assertions or testbench monitors. The replay technology can then be

simulated "normally" for the time-window in question while dumping complete simulation event data. In effect, in this flow, the problem has been broken down and localized to a time-window and the enabling technology for this technique utilises the natural behavior of circuits to settle to a deterministic state when only registers and primary I/O are applied (forced) at a particular time.

As with essential signal analysis and data expansion, the replay techniques depend on significant technology under-the-hood but the user-experience is minimally impacted. For replay, the user can view the simulator as a server that can be queried to provide accurate timing results for any user-specified time-window. In fact, the technology allows the simulator to stay "alive" (as a server of sorts) with the user asking for arbitrary time-windows in any order to be simulated with recording. The multiple mini simulations are performed with the same essential signal dump recorded earlier without ever exiting the simulator run.

Using Replay techniques, benchmarks show significantly faster simulation times and 10X smaller dump file sizes when compared to traditional methods, while also providing the accurate results needed to debug timing issues that arise during chip implementation. Combined with essential signal and data expansion techniques, Replay can provide a complete solution that optimises verification flows without sacrificing visibility.

In the future, the technology can be extended for other applications such as targeted RTL simulations and reducing the turnaround for verifying fixes.

SpringSoft | www.springsoft.com
Bindesh Patel is a technical marketing manager at SpringSoft

It's all in the timing

Timing is everything when it comes to chip design.
Bindesh Patel looks at timing closure

computation is localised and optimised for the typical tasks undertaken by a human engineer to debug and analyse bugs in the design. In effect, users see no effect on their debugger usage but gain significant simulation time and file savings with no loss in visibility.

A key trait of the data expansion engine is that it is purely functional and computes values on cycle boundaries; in effect, timing is ignored. For gate-level designs, however, timing may be required to debug and isolate specific bugs.

Gate level timing

Visibility enhancement can be augmented with Replay-based techniques to address issues discovered by gate-level simulations that include timing. The goal is that engineers should be able to literally "replay" short, quick simulations focused on only problematic time windows observed during simulation. The significant technology employed here highly leverages the aforementioned optimised simulation dump that only includes a relatively small set of essential signals. A flow applying this technique may include, but is not dependent on, a first-pass debug using the essential signal dump and data expansion. This initial pass may be used to narrow down the time

utilized to facilitate fast, targeted simulations to capture full signal data with timing for the time windows in which problems originate. The efficiency gains are realized by instructing the simulation to jump to the beginning of the user-specified time window, eliminating the long run-time usually spent getting to that point. To enable the subsequent accurate operation of the simulation model, the previously captured essential signals for that time are applied (forced) and the circuit is then

