# Gatelevel Simulations

## Continuing Value in Functional Simulation

Ashok Chandran, Project Manager, Analog Devices, Inc., Bengaluru, India
(*ashok.chandran@analog.com*)

Roy Vincent, IC Design Engineer, Analog Devices, Inc., Bengaluru, India
(*roy.vincent@analog.com*)

***Abstract*—Gatelevel simulation (GLS) is an integral part of the chip design flow. With growing digital complexities of SoC designs, the effort involved in getting test-suites running through GLS has been increasing exponentially. The paper highlights the value proposition of continuing to invest in GLS - the issues that get caught through gatelevel verification, why these do not get caught in any other verification/implementation flows, key considerations and the challenges.**

***Keywords*—GLS, SoC, Netlist, SDF**

## I. Introduction

Gatelevel verification is a standard industry practice as the sign-off in chip design flow. It is closer to the real silicon behavior and provides high level of confidence on functionality and timing compliance. It is performed by annotating timing information from SDF (standard delay format) onto the post-route netlist. This forms the most useful abstraction level for running extensive functional tests (as transistor level simulations are very slow at SoC scale) to validate timing.

## II. Typical Flow

Gatelevel simulations are carried on the Slow and Fast corners as identified by TA. The flow involves several edits to the SDF and Netlist to speed up simulations and add checks. The libraries are also modified to account for variations in timing check behavior.

Over and above using GLS as a confidence measure, we add checks and utilize it for bug-hunting. Examples of such checks:

- Adding synchronizer input glitch checks.
- Injecting random o/p on synchronizer misses to catch CDC issues.
- Pessimistic checks for reset recovery violations.
- Random initialization to non-reset flops which are flagged as fine by designers.
- Adding pessimism on pulse rejection values to enable glitch propagation.
- AC Timing checkers which checks for o/p timing and inject x on inactive window of inputs.
- Enabling Timing models for all memories.
- Inserting timing checks on Macro inputs.
- Cycle compares between RTL and GLS to catch any behaviour mismatch.

## III. Value

The below are the key issues found through gatelevel simulations. Some of these have alternate checks or practices that can be done earlier in the design cycle.

- X-Pessimism: For flops which use synchronous reset or are non-resettable, the value being 0/1 on power-up may result in different behavior. RTL simulations may mask off the side-effects of such uninitialized flops. Such issues could be potentially fatal functionally and are visible via gatelevel simulations.

- Timing Constraint Issues: These are typically wrong multi-cycle/false path constraints in the timing analysis deck. This could also be due to missing constraints as well.
- Reset Domains: When different blocks in a chip have different reset domains, it becomes important that the reset of one domain does not violate timing on the interface flops of the out of reset domain. Such reset paths are not checked in Timing Analysis by default.
- Glitch Issues: Combination glitches on certain macro inputs (like memories or 3rd Party IPs) may be potentially fatal. These are never visible in RTL simulations and may miss checks in the Timing Analysis deck if not defined properly.
- Combination logic on Synchronizer Inputs: This could result in false capture of data and can get caught in Gatelevel Simulations. This is caught via additional checks inserted in GLS.
- AC Timing: The external pin timing specifications can be cross checked against TA deck through gatelevel simulations by enabling timing checks and x-drive on inputs.
- Reset Glitch: Glitch on reset inputs may cause unintended resets to happen. These are again not visible in RTL simulations. However, these can be caught via checks in TA.
- Reset Recovery: Changing of D input and Reset of flop together causes a race. The behavior may be non-deterministic and flagged via adding pessimism in checks.
- Library issues: All timing at macro inputs are checked via inserted timing checks. This validates the TA checks from Libs and sometimes highlight mismatches in Lib against functional expectations.
- LEC / Formal Issues: If formal equivalence is done without using all defines as in simulation, we may end up with the case of simulation vs synthesis mismatch. Typically we have seen that `ifdef SYN misses some connectivity which was present in the other case.

## IV. Challenges

There are several challenges with respect to gatelevel simulations:

- The simulations are very slow due to large number of cells, delay annotations and timing checks. Compiles are also slower due to the same. Simulation debug is also slower due to the same.
- Consume significant machine memory and hence limits the number of batch simulations possible.
- As taps from design may get optimized off, you have to fix testbench checkers/assertions for the gatelevel simulations. This consumes significant time and is an iterative process.
- The effectiveness of GLS is seen when running functionally qualified tests. Arriving at large number of such tests is a really challenging task.

## V. Approach

Gatelevel Flow improvements done in recent projects:

- Tap preservation flow to enable faster gatesims deck bring-up. This was done through simulator options to list down taps into the design.
- LEC/Simulator Script based flow to generate list of inverted taps to avoid debug iterations.
- Extensive use of TCL interfaces in simulator to speed up iterations by avoiding need to re-compile.
- Separate compile of design and testbench for speeding up compile while working on testbench edits was also tried.
- Full automated flow/scripts for doing all SDF/Netlist hacks matured over multiple projects.

- Ability to run full random simulations at fullchip level on GLS.

## VI. Results

With efforts put into in Gatelevel verification for our projects, the value has been found to be very high and justifies the time investment in bring-up of the simulation deck and regressions. We have found that GLS acts as a good quality analysis for the functionality and timing and improves confidence in the design.

## VII. Summary

As highlighted in the introduction, there are several issues in implementing a gatelevel simulation flow. Some of these can be addressed early in the design cycle. There are several options for easing the bring-up of gatelevel deck by trading off in synthesis/PNR. Tap preservation flow was tried out and provided some improvement. We also developed flows for identifying missing taps via logic equivalence checks. There is scope to work with tool vendors and get more robust flows.