Addressing Renewed Gate Level Simulation Needs for 10nm-28nm and Below

Gagandeep Singh, Cadence, Noida, India (gagans@cadence.com)

***Abstract*— Gate level simulation is critical in the verification cycle but often overlooked for newer challenges such as test bench development with the UVM. However, the increase in design sizes and the complexity of timing checks seen in technologies nodes 10-28nm and below are responsible for longer run times, high memory requirements, and a growing set of GLS applications including design for test (DFT) and low-power. Also GLS can catch issues that STA or logical equivalence tools are not able to report. Continuous improvements have been on going in the simulator for last few years to improve the Out of box gate-level simulation performance by adding optimizations in the simulator. However, in order to match the verification requirements for newer, larger designs, that have increased complexities, a combined simulation and methodology approach has to be taken.**
This paper captures some newer, innovative techniques that help increase the effectiveness of the gate level simulation flow, along with some of the best practices generally used by verification teams.

Keywords—Gate level simulation methodology, Timing verification, Efficient Gate level simulation

I. INTRODUCTION

This paper describes new methodologies and some best practices that increase GLS productivity with timing in addition to the continuous Out of Box simulator performance. In technology nodes 10-28nm and below, it is seen that GLS with timing is around 6-7x more expensive than GLS in zero delay mode due to complex timing checks and their conditions. Also GLS in zero delay mode is around 1.5x-2x slower than RTL. So running GLS efficiently can save a lot of verification effort. This can be done by using innovative methodologies based on the information available with static tools like static timing analysis, logical equivalence and passing that information for efficient gate level simulations. Such techniques can help designers focus on the verification of real gate level issues and not the ones that have already been caught with other tools in some way.

Section II captures all the GLS Methodology techniques and gains seen in some real designs.
Section III is the conclusion and followed by references, Section IV

II. GLS METHODOLOGIES

1. Hybrid Mode (RTL+GLS)

Big, complex SoCs are made up of multiple IPs and in order to verify the each gate-level IP at SoC level, a hybrid mode of simulation can be used. Also different IPs in the design come at different phases of the design cycle so GLS netlist can be used for the blocks that are available and need to be tested, while use RTL for the other portions of the design. Since RTL runs much faster and will take much less memory, especially in technology node 10-28nm and below, the verification of the IPs at GLS level with/without timing can be done easily. This can give a huge run time improvement and keep focus on verifying selective GLS IPs at a given time. In addition to performance benefits, hybrid technique also enables GLS much early in the design cycle so can save a lot of verification time.
Please note: For functional issues, GLS should run in zero-delay mode while for verifying the timing, GLS with timing should be used.

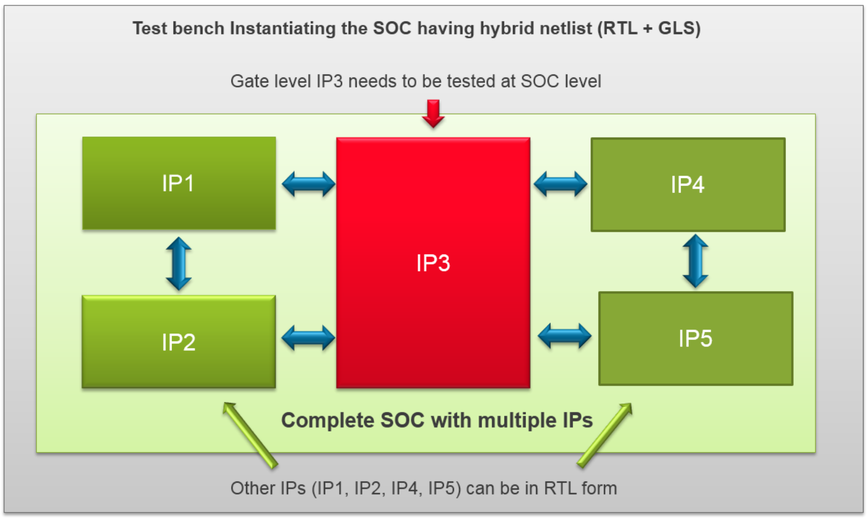

Test bench Instantiating the SOC having hybrid netlist (RTL + GLS)
Gate level IP3 needs to be tested at SOC level
IP1
IP3
IP4
IP2
IP5
Complete SOC with multiple IPs
Other IPs (IP1, IP2, IP4, IP5) can be in RTL form

Figure 1: Hybrid RTL+GLS

Performance Gains seen

Hybrid GLS + RTL mode can give a huge productivity benefit and can reduce upto 2xGLS verification time as different IPs come at different intervals of time. This also reduces the heavy debug effort to debug any functional issue or timing issue at the IP level. When all the IPs are functionally/timing verified and then running full GLS at SOC has potentially lesser number of issue. It can help designers to resolve the integration level issues.

2. Using abstracted timing models for GLS with timing

Abstracted timing model (patent pending) can be generated using STA tool that keeps the interface timing only and can be used to verify the timing at complete SOC level

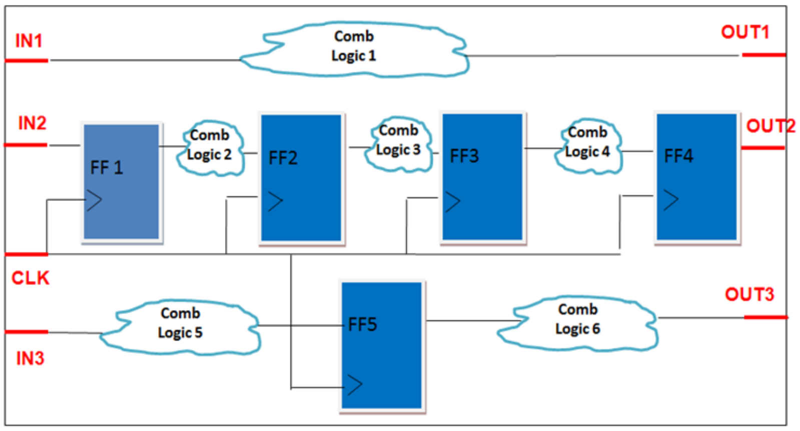


Figure 2: Original Design

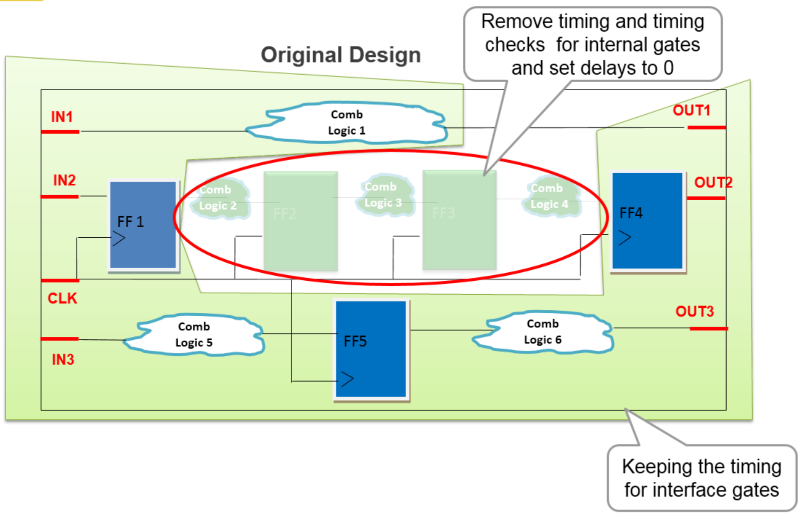


Figure 3: Abstracted timing model (Interface timing)

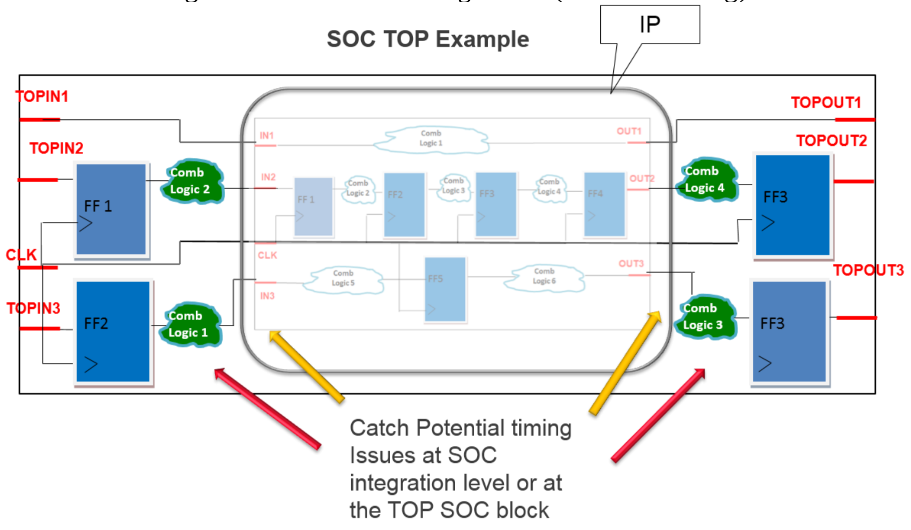


Figure 4: Integration at SOC level

The abstracted timing model, removes timing for internal flops & gates for IPs that are either fully Verified by STA or GLS with timing verified at IP level using IP level tests or Verified with GLS+RTL hybrid simulation. It preserves the interface timing accuracy at the port level and gives better performance in GLS as only limited/required timing is enabled.
As for lower technology nodes timing & timing checks are very expensive so for huge SOCs, it is nearly impossible to run GLS with timing due to much higher run time and machine memory limitations. So abstracted timing model, allows verification engineers to run GLS with timing at the SOC level without caring about the internal timing of different blocks that have been already verified.

Figure below shows the timing of abstracted timing block matches with full timing at the port level

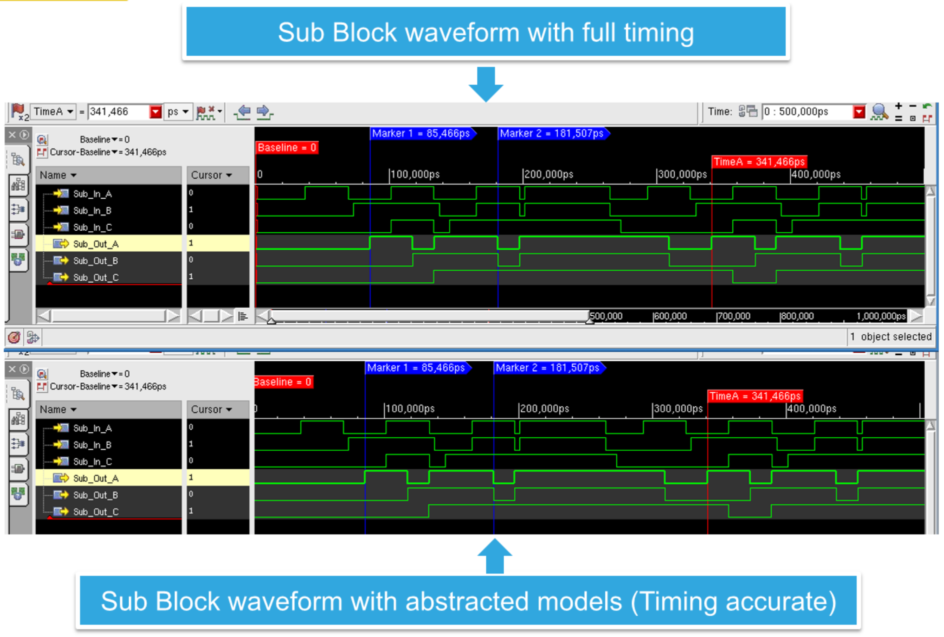


Figure 5: Port level timing abstracted timing model vs full timing

Performance Gains seen

Abstracted timing model has been tried on some design and has shown promising results. It has helped reducing the simulation run time around 1.5x-2x and reduce memory around 1.25-1.5x depending upon the level of abstraction in the design.

3. Verifying long GLS initialization through hardware accelerator

Gate-level timing simulations for large SoCs have long runtimes as they have complex timing checks, and it can take several days just for chip initialization. In addition to the long runtime, the other key challenge is debug of GLS environment given that the turnaround time increases significantly with each iteration. Bring-up tests are typically run separately to verify the bring-up and initialization of the SOC. For all the other tests, the timing or timing check during the initialization/configuration phase is of no significance/interest as it is tested with bring-up tests. So a unique (patent-pending) concept of Simulation- Acceleration methodology/flow with sdf can be used where the sdf annotation is honored when the run is in software simulator and without timing when the run is on the emulator.

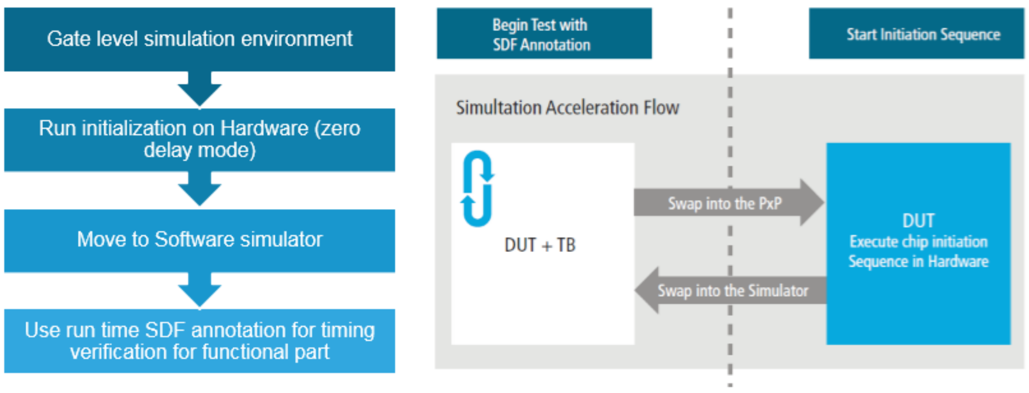

Gate level simulation environment
Run initialization on Hardware (zero delay mode)
Move to Software simulator
Use run time SDF annotation for timing verification for functional part
Begin Test with SDF Annotation
Start Initiation Sequence
Simultation Acceleration Flow
DUT + TB
Swap into the PxP
Swap into the Simulator
DUT
Execute chip initiation Sequence in Hardware

Figure 6 Hot-Swap

With this methodology, the netlist simulations can be run with high performance (zero delay mode) on hardware by swapping the design into the emulator for the complete initialization/configuration phase. Then, swapping out of the emulator once initialization is complete and running the rest of the test in the software simulator with sdf. The simulator has a capability to annotate SDF during run time. So this feature is used once the initialization is complete and simulation moves to the software side.

Performance Gains seen

In some of the designs the initialization itself was taking more than 24hrs and remaining simulation just ran has 3-4hrs. Using this approach the overall simulation was completed in 6-7hrs as initialization in zero delay mode completed much faster on the hardware box.

4. Saving and Restarting Simulations

Typically, much GLS time is spent in the initialization phase. This time can sometimes be very significant. As a result, a single simulation should be saved and all other (n-1) simulations should be run from the saved checkpoint snapshot.

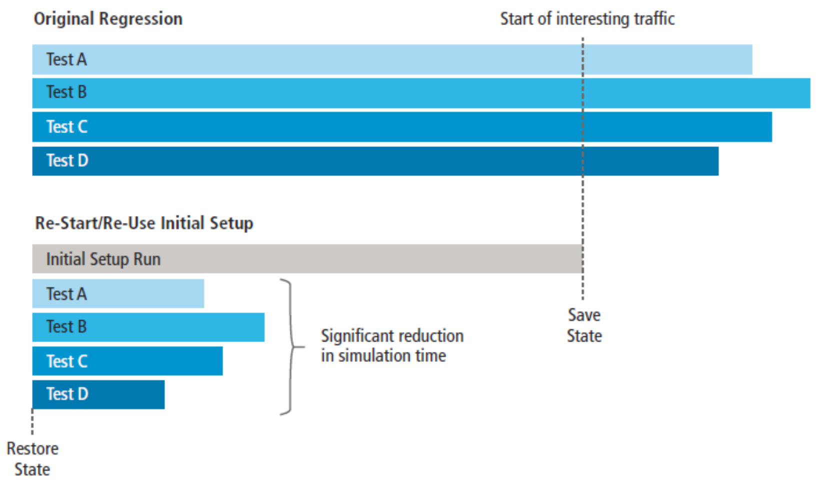


Figure 7. Save Restart

Performance Gains seen

Initialization sequences are generally quite long and save restart methodology has seen upto 60% improvement in simulation run time by re-using the sequence.

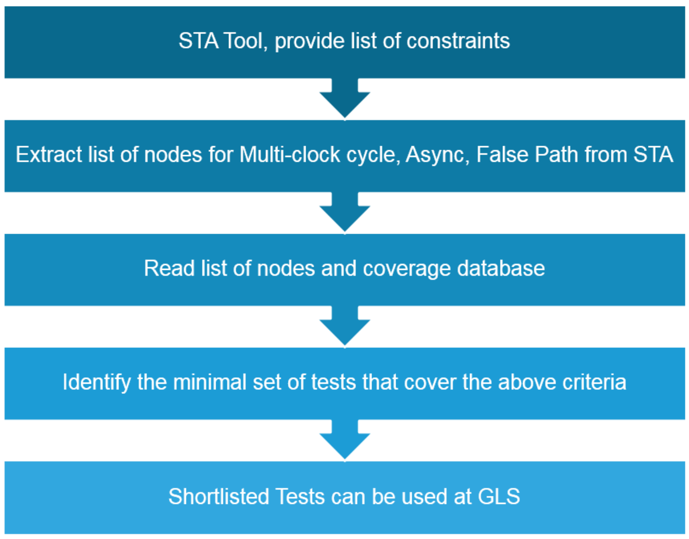
5. Minimum Tests required at GLS timing based on RTL tests.

Static timing analysis tool is not able to catch all the issues that can only be seen in a GLS run. There are some assumptions or constraints written in STA that can be confirmed only during GLS. Some of the typical scenarios which STA is not able to cover.

- STA constraints verifications (MCP, False, Maxdelay etc) to ensure the assumptions are correct. Constraints might be written incorrectly by the user
- Inability of STA to handle asynchronous paths.
- Functional Glitches detection related to timing

- Clock related: Clock domain crossing for asynchronous clocks, Clock Gating for low power, Dynamic changing of clocks
- Low power mode, switching on/off power domains with timing
- Verify power up and reset operation of the design with timing

STA tool GLS with timing tests should focus on limitations of STA tool as mentioned above. It is not required to run all the RTL tests at GLS and currently identification is done manually based on verification engineer experience. The test identification can be automated and the flow will identify tests that can potentially find these issues. These tests must be run first to verify timing and more can be run later in case time resources and time is available.



STA Tool, provide list of constraints
Extract list of nodes for Multi-clock cycle, Async, False Path from STA
Read list of nodes and coverage database
Identify the minimal set of tests that cover the above criteria
Shortlisted Tests can be used at GLS

Figure 8. GLS with timing test identification

Performance Gains seen

It is generally seen that 10-20% of the RTL tests are sufficient for running GLS with timing. This is based on the initial trials done using the above flow on a few designs. This data is also matches with the survey done with different verification engineering teams who select the test manually. They typically run between 10-25% of their RTL tests at GLS.

6. Controlling timing checks based on STA report

Since STA does the complete timing analysis and is good in handling synchronous paths. In cases where timing for the complete or a portion of the design is already met and the paths are synchronous, the timing checks during simulation might not be required for this portion of the design, specifically the internal flops.
The STA and simulator flow is illustrated in Figure 9 and STA can remove the timing checks of synchronous areas of the design that do not have complex timing constraints in STA.
Please note: Here only the timing checks (setup, hold, recovery, removal, width etc) are switched off, the timing i.e. iopath, interconnect, port delays etc are honored so design still runs with timing.

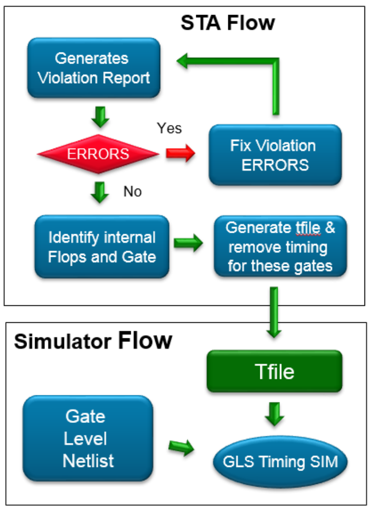

STA Flow
Generates Violation Report
ERRORS
Yes
No
Fix Violation ERRORS
Identify internal Flops and Gate
Generate tfile & remove timing for these gates
Simulator Flow
Tfile
Gate Level Netlist
GLS Timing SIM

Figure 9: Controlling timing checks based on STA

Based on STA, timing check of internal flops can be switched off. In Figure 10, a timing file to control timing checks be generated for flops marked in the grey as they are internal

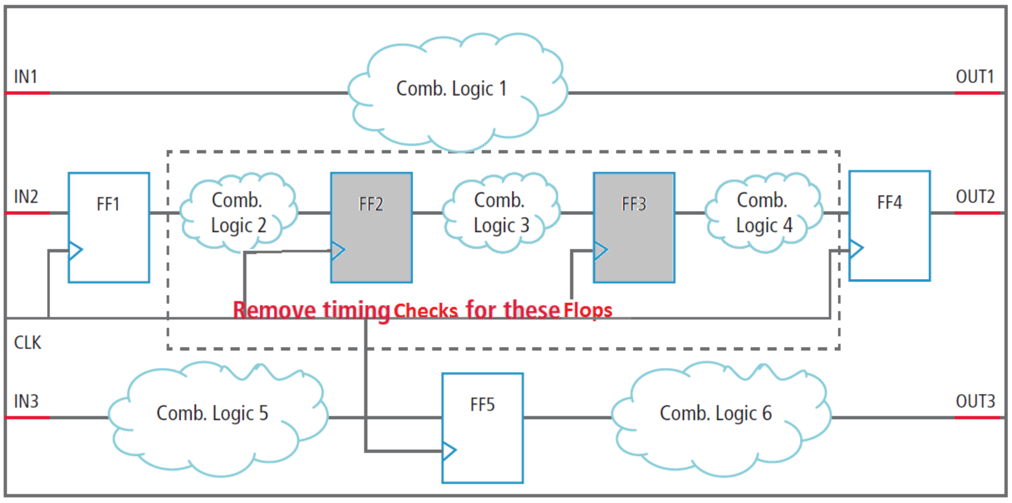


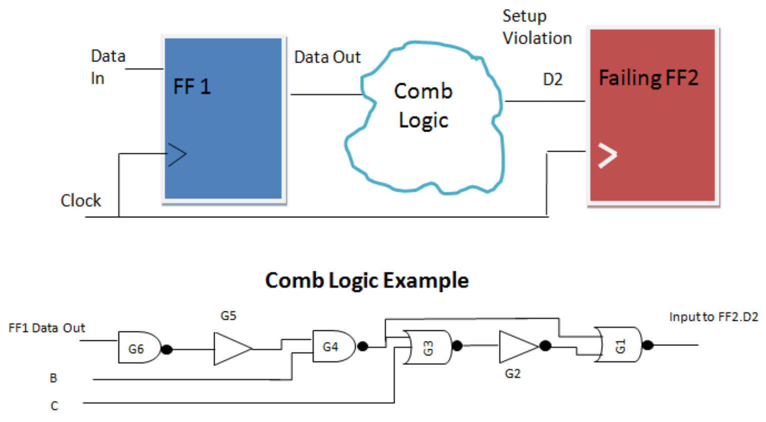
Figure 10: Remove timing checks for internal flops

Performance Gains seen

In technology nodes 10nm-28nm or below, it is seen that timing checks have run time impact of 2-2.5x and require 1.25-1.5x more memory. So if good number of timing checks are removed, it can show a significant improvement in run time and memory and still making the design run with timing.

7. Starting GLS early even if STA timing is not completely clean

There can be scenarios where the design might not be complete timing clean but majority of it is timing clean. However in order to run meaningful GLS with timing, it is important that the design is timing clean so that verifications engineers do not spend unnecessary time in debugging the issues already reported by STA.
A flow in STA environment can be written that fixes the delays (setup & hold violations) temporarily and dumps timing correct SDF that can be used in GLS to catch the other potential issues that are seen at STA. In the meantime, timing issues can be fixed in parallel by STA team.



End Point	Slack (ns)	Cause
Top/core/ctrl/FF2/D2	-0.56	Violated

Figure 11: The example shows setup violation reported by STA.

Results seen

GLS early even if the complete timing closure is in process has been tried on customer designs.
Design 1: There were total of 724 setup violations reported by STA and required fix in the design. All 724 setup violations were successfully removed, ignored in generated SDF to help start Gate level simulation early.
Design 2: There were total of 68 setup violations and all 68 violations were successfully removed in generated SDF.
The technique helps removing, ignoring timing violations reported by STA tool and dump out SDF file that has no violations for GLS simulation. And in parallel, the actual timing issue in the design can be fixed by STA team. This helped designer to run meaningful GLS with timing simulation early in design cycle without spending unnecessary effort in debugging the issues that are already reported by static timing analysis tools like STA.

III. Conclusion

Continuous improvements are seen in the simulation and they are still ongoing. However, in order to match the verification requirements for newer, larger designs with increased complexities, a holistic approach is required. A combined simulation and methodology approach has to be taken in order to achieve an effective and efficient verification process

IV. References

GLS white paper - https://www.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/system-design-verification/gate-level-simulation-wp.pdf