Static Sign-Off:
6 Fundamentals to Maximize Design & Verification Efficiency

by Prakash Narain, Real Intent President & CEO

I. Overview: Static Sign-Off during Design Process Closes Key Gaps

Changing global design strategies needed to deal with larger designs are driving increased usage of **static sign-off during the design process** to improve design & verification efficiencies. The use of sophisticated clocking, test and power management approaches create a new set of challenges that must be addressed.

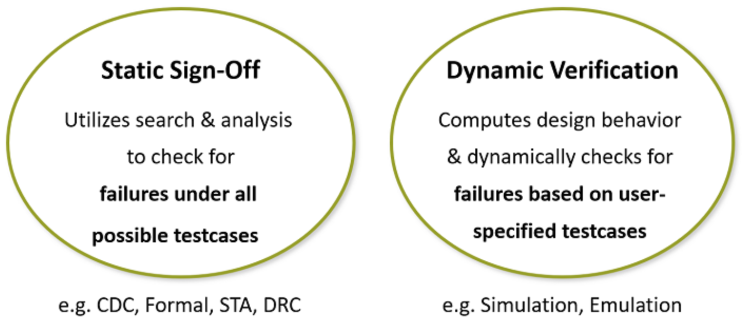
For targeted domains such as CDC and RDC, traditional verification is inefficient at scale, while deploying static methods with customized analysis during design process can deliver fast & complete sign-off for multi-billion gate designs.

This paper discusses six fundamentals to understand about static sign-off, including target domains, how it compares with simulation and formal against key metrics, selected implementation practices, and how static sign-off tools fit with design methodologies.

II. Static Sign-Off vs. Dynamic Verification

How do Static & Dynamic Methods Differ?

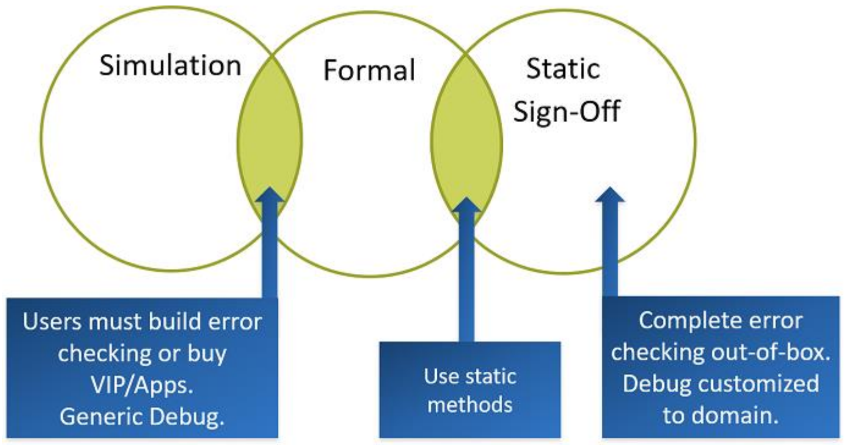
Dynamic methods compute design behavior for user specified test cases, and then check the computed behavior for failures. Simulation and emulation are good examples.


Static Sign-Off
Utilizes search & analysis to check for **failures under all possible testcases**
Dynamic Verification
Computes design behavior & dynamically checks for **failures based on user-specified testcases**
e.g. CDC, Formal, STA, DRC
e.g. Simulation, Emulation

In contrast, static methods utilize search and analysis techniques to check for design failures under all possible test cases. At the broadest level, static sign-off technologies span layout (DRC), timing (STA) and functional (CDC, RDC, X-Propagation…). This paper will focus on functional static-sign-off.

Comparing Simulation, Formal & Static Sign-Off

To help further understand the role of static sign-off, let's compare it with simulation and formal verification.


Simulation
Formal
Static Sign-Off
Users must build error checking or buy VIP/Apps. Generic Debug.
Use static methods
Complete error checking out-of-box. Debug customized to domain.

Formal and simulation are both general applications. It's the user's responsibility to build error checking – they can do so by purchasing VIP or apps. Debug is also generic.

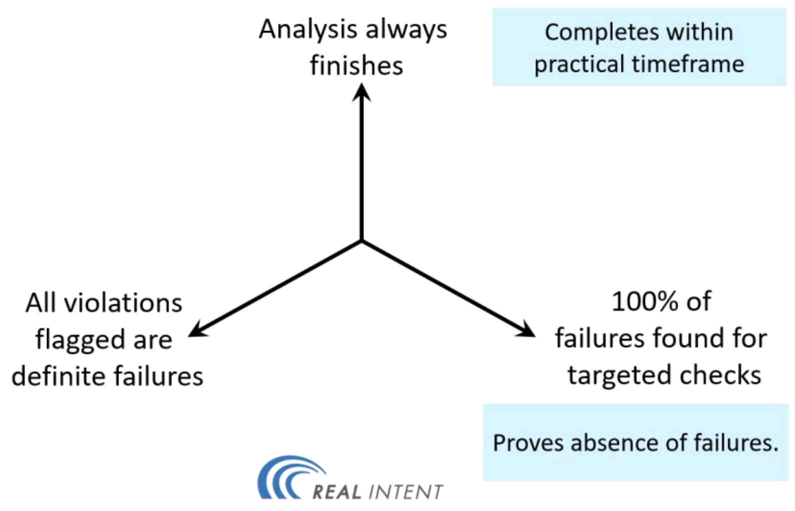
Static sign-off and formal are both static approaches. However, they differ in that static sign-off tools include customized and complete error checking, which allows debug to also be customized to the specific application.

III. Six Static Sign-Off Fundamentals to Maximize Verification Efficiency

1. Use Verification Metrics to Assess Approach Efficiency

A simplified view of three key verification metrics to evaluate the merits simulation, formal and static sign-off are:

1. The analysis always finishes — meaning that it completes in a practical time frame.
2. All the violations flagged by the analysis are definitely design failures, vs. also including 'warnings' of potential failures.
3. 100% of the failures are found to the targeted checks. Thus, the analysis proves the absence of any design errors.


Analysis always finishes
Completes within practical timeframe
All violations flagged are definite failures
100% of failures found for targeted checks
Proves absence of failures.
REAL INTENT

As you can see, an ideal verification methodology would rate a perfect score on each metric. However, there is no ideal methodology today; each functional verification approach has its strengths and weaknesses.

Below we can see how the various methodologies measure against these metrics, and where engineering teams spend resources overcoming the deficiencies.

Verification Metric	Simulation	Formal	Static Sign-off
Analysis always finishes	Yes	No	Yes
100% of failures found for target checks	No	Yes	Yes
All violations flagged are definite failures	Yes	Yes	No

Simulation

Simulation always finishes – this is because when you create a test bench, you understand about how long it will take for simulation to run before commencing (for larger designs, you may deploy emulation). Additionally, all failures flagged by simulation are definite failures.

Simulation's weakness is that it cannot confirm whether 100% of failures have been detected. Thus, engineering teams spend a lot of effort estimating coverage and writing test cases to achieve coverage of these imperfect targets.

Formal Verification

With formal analysis, all failures that are flagged are definite failures. Formal is also capable of finding 100% of the failures.

However, formal's deficiency is on performance dimension; formal analysis may not finish in the project timeline. The result is that a lot of engineering effort goes into trying to improve the completion rate.

This is where highly trained experts work with the tools to maximize completion of formal analysis.

Static Sign-Off

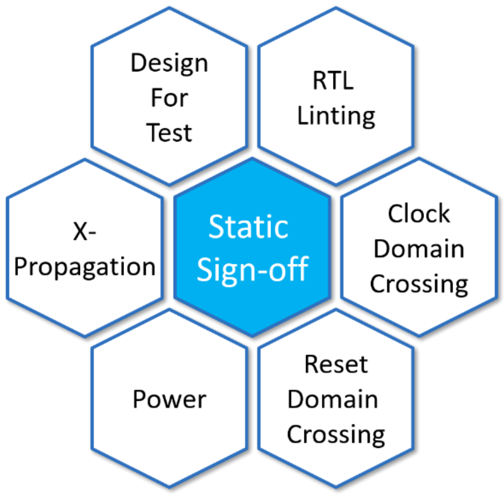
Static sign-off always finishes. It also finds 100% of the failures targeted by the check. However, with static sign-off, all violations flagged are not definite failures, the results have some "noise".

The noise level for the static sign-off tools will vary between tools and vendors, which impacts the level of engineering effort required.

Thus, static sign-off is best deployed for targeted domains where all errors must be identified, and the analysis must complete.

2. Deploy Static Sign-Off across Multiple Target Domains

There are multiple domains where development teams can efficiently apply functional static sign-off today. Let's look at how static sign-off is deployed for each of these target domains.



Static Sign-Off Target Domains

- **RTL Linting.** Enforces coding guidelines and identifies functional issues prior to simulation. Rules cover syntax, semantic, and style checks, to ensure high quality RTL.

- **Clock Domain Crossing.** Structural and functional analysis ensure that signals crossing asynchronous clock domains are received reliably, verifying that the data will be transferred across clock domains without introducing design problems.

- **Reset Domain Crossing.** Confirms that signals crossing reset domains function reliably, identifying metastability problems and glitches arising from software and/or low power asynchronous resets, or re-convergence of synchronized resets.

- **Power.** Ensure that power management is correctly implemented, and power switching will cause the chip to malfunction.

- **X-Propagation.** Performs audit and debug of design initialization to identify potential X-optimism and prevent inaccurate RTL simulation.

- **Design for Test.** Early RTL and netlist analyses are performed to identify all testability violations in a timely and efficient manner before full DFT/ATPG.

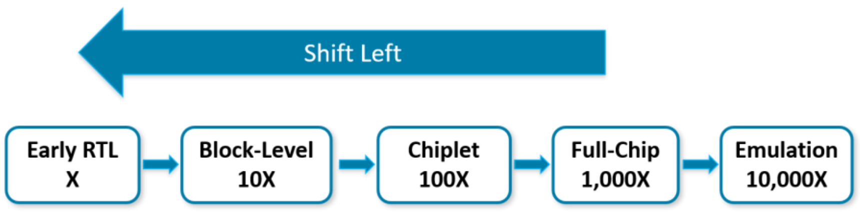
Most chips have at least three or more targeted domains where static sign-off is used effectively. The domains will vary depending on the design and priorities.

Nvidia provides examples of the broad range of static checks that they use in their design process.

3. Shift Left — Start Static Sign-off Early

It is well-accepted in verification that the earlier you can find and fix bugs, the more cost effective it is.

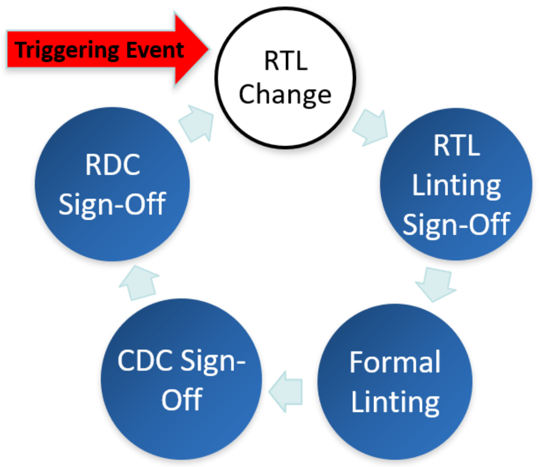
In fact, bug fix costs generally go up 10X at each stage. As we can see here, emulation, while important, occurs so late in the design, that any bugs found there are much more expensive. Thus, the move to 'shift left' for verification.



Thus, static analysis is best begun as early as possible. Fujitsu discusses their 30 percent engineering time reduction resulting from deploying early static analysis.

4. Execute Static Checks "Continuously"

As more companies begin static checks early, they are also deploying continuous sign-off throughout RTL. This is very effective in minimizing the sign-off burden.



Continuous Static Sign-off Checking

Any changes in RTL can be used to trigger a new cycle of static checks in parallel.

Google gives a good example of deployment of continuous static sign-off checks.

5. Ensure Your Static Sign-off Methodology is Complete

True static sign-off can't miss failures. An example of this is single-mode clock domain crossing RTL sign-off, which has been in mainstream use for many years now.

However, a "complete" clock domain crossing sign-off goes beyond this. To catch all CDC errors, some companies will need to deploy:

1. Single mode and Multimode CDC
2. RTL and gate level CDC

Multimode CDC covers all scenarios in a single run. This can be critical for companies with a lot of modes using single mode CDC, as otherwise they typically only run selected modes due to deadlines. They can miss failures by skipping modes.

Further, RTL CDC sign-off isn't always enough. Logic synthesis can insert new registers and power reduction may add clock gating cells, both of which can introduce new CDC errors at the netlist level in some designs. Even timing optimization can change the logic organization. This is where gate-level CDC can be necessary.

6. Align Static Sign-Off Methodology with Tools

There are common practices for static sign-off, but there is no default methodology today.

The overall goal here is to invest in and establish a static sign-off methodology and tool set that will minimize the one deficiency with static sign-off: the noise level. Doing so will reduce manual engineering effort and total project time for completions.

Aligning your static sign-off methodology with your tools can be achieved as follows:

1. Start by defining your methodology.
2. Define your static tool checks to enforce the methodology (Lint policy, CDC rules, RDC scenarios, X-analysis policy, DFT rules…)
3. Refine your static configuration based on your initial results to reduce noise.
4. Work with your tool vendor to refine the tool to align with your methodology. (rather than vice versa). This requires tool and vendor to be flexible and highly responsive.
5. In parallel, enable strong debug and potential continuous integration.

Through this practice, Real Intent has reduced noise level for our static sign-off tool, and thus improved our customer’s engineering resource utilization and project completion schedules.

One recent example of this is our work with Samsung on their hierarchical CDC methodology, which reduced the number of CDC violations by 95 percent, and decreased their engineering effort by 70 percent.

IV. Conclusion

The use of static sign-off during the design process continues to expand as development teams experience its effectiveness.

Considering these six fundamentals as you evolve your team’s design methodology can help to you better optimize your design & verification efficiency.

About the Author

Prakash Narain, President & CEO of Real Intent

Dr. Prakash Narain shifted Real Intent's technology focus to Static Sign-Off to better tackle verification inefficiencies back in 2010. Since then, Prakash has expanded this effort to now offer six tools focused specifically on different static sign-off domains, ranging from CDC to RDC to Linting — with more coming.

His career spans IBM, AMD and Sun where he had hands-on experience with all aspects of IC design, CAD tools, design and methodologies. He was the project leader for test and verification for Sun's UltraSPARC III, and an architect of AMD's Mercury Design System. He has architected and developed CAD tools for test and verification for IBM EDA.

Dr. Narain has a Ph.D. from the University of Illinois at Champaign-Urbana where his thesis focus was on algorithms for high level testing and verification.

###