# A Faster and Efficient Timing Constraint Verification Methodology for GFX SOCs

Vineeth B, Deepmala Sachan
Intel Technology India Pvt. Ltd.
Bengaluru, Karnataka
*{vineeth.b|deepmala.sachan}@intel.com*

***Abstract*- It is of the utmost importance to ensure that timing constraints for GFX SOCs are accurate for proper synthesis results and timing closure so that the design meets the desired performance requirements. A gate-level simulation (GLS) is traditionally used to verify timing constraints. Simulations such as these require long run times, offers less coverage, occur much too late in the design cycle, and require a lot of effort to debug failures, which results in a longer turnaround time (TAT) to uncover and fix timing constraints issues. The paper presents a methodology for verifying timing constraints at an earlier stage of design cycle that is faster and more efficient. The proposed methodology includes constraint linting to identify clock propagation issues and unmapped constraints, formal verification of timing exceptions, generating assertions corresponding to exceptions that fail formal verification, and verification of the assertions in simulation by running the entire regression suite to identify incorrect timing exceptions.**

***Keywords—RTL design; Timing constraints; Timing exceptions; Design constraints; Assertions; Simulation;***

## I. Introduction and Background

GFX SOCs are complex, high-performance designs with multiple clock domains, numerous functional units, and intricate interactions between different modules that often involve timing dependencies. The different IP modules can be coming from multiple internal as well as external vendors and would be delivered along with their respective timing collateral. However, there is no guarantee that these constraints would hold true when the IPs are fully integrated into the final SOC as per the design requirements. Therefore, timing constraint verification is very crucial to ensure design stability by identifying potential timing violations and avoiding issues such as data corruption, race conditions, and functional failures.

The conventional method to verify timing constraints is gate-level simulation (GLS). In GLS, the gate level netlist of the design along with respective timing information is simulated using proper testcases to identify any potential timing issues. However, such simulations require long run times and often offers less coverage [1]. Also, it occurs too late in the design development cycle and requires lot of effort to debug the failures. This makes it very difficult to close all the timing violations within the scheduled project timeline and can lead to potential silicon escapes and costly re-spin of the silicon. Therefore, the several disadvantages of GLS dictates the need for a faster and efficient methodology for verifying the timing constraints.

The previous efforts include strategies [2] intended to lessen the risk and effort required for GLS while verifying timing constraints based on the shortcomings discovered during numerous projects. However, these techniques do not provide a generic solution and only alleviate some of the disadvantages of GLS. The methodology presented in this paper, on the other hand, provides a quick and effective method for verifying timing constraints, overcoming the limitations of GLS-based verification, and considerably reduces the time required to find and resolve all the timing constraint issues, thereby providing a significant shift-left in the overall timing closure of the design. Section II describes the formal verification methodology, followed by timing exception assertion verification methodology in section III. The results are presented in section IV based on which paper is concluded in section V.

## II. Timing Constraint Verification

### A. *Proposed Methodology*

The proposed methodology for timing constraint verification is depicted in Figure 1. The primary inputs are RTL file list, HIP collateral, timing constraints, design constraints and the block level configuration file. The HIP

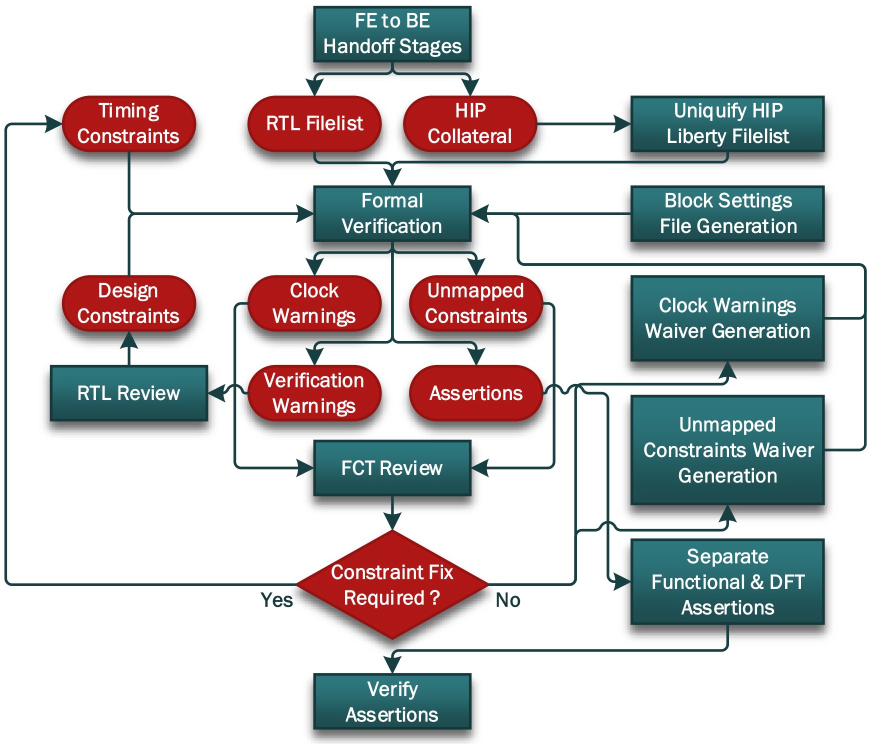

Figure 1: Timing Constraint Verification Methodology

collateral has liberty files corresponding to multiple corners and scenarios for each HIP module and hence if directly used as input to the constraint verification flow, it can lead to crashes due to memory issues. To avoid such issues, the HIP collateral must be uniquified by maintaining only a single liberty file for each HIP module before giving as input to the flow. The timing constraints are provided by the full chip timing team (FCT) team. The block level configuration file has all the flow options as well as all the TCL variables that are required to properly source the FCT delivered timing constraints.

During timing constraint verification, the constraints are first mapped to RTL and all the unmapped constraints are reported. These can be mainly due to following two reasons:

1) Syntax issues
2) Hierarchy mismatches between RTL and netlist

All the syntax issues must be fixed, and the hierarchy differences should be resolved by providing a mapping file which maps corresponding hierarchies in RTL with netlist. After the constraint mapping is done, several clock linting checks are performed to identify issues with clock propagation in the design. All the clock warnings should be reviewed, and the critical issues should be fixed in constraints and appropriate waivers should be added for the rest of the violations. Once all the unmapped constraints and clock propagation issues are resolved, formal verification of timing exception is done, and assertions are generated for all the exceptions that fails the formal verification. The generated assertions are then separated into functional and DFT assertions and must be verified in functional and DFT simulation regressions respectively.

## B. *Formal Verification Improvements*

There are many scenarios where the flow may require additional inputs to formally verify a timing path. In the absence of such inputs the number of formally failing exceptions as well as the corresponding assertions generated would be large and consequently requires huge effort to validate them in simulation. The different methods to improve formal verification of timing exceptions are discussed below:

1) Specifying the list of synchronizer cells in the design: The formal verification passes for all timing paths having synchronizer cell as the end point.
2) Specifying constraints on input ports and asynchronous reset ports impacting formal verification based on the verification warnings review:
    a. Input ports must be constrained to its legal values.
    b. Asynchronous reset ports must be constrained to its non-reset value.

    In the absence of such constraints, the tool may propagate more clocks and constants throughout the design than is appropriate for the constraints being verified.
3) Applying constraints from Clock Domain Crossing (CDC) analysis to eliminate non-functional (DFX) paths from verification.
4) Adding specific endpoint waivers based on RTL feedback where it is ensured by qualifiers that endpoint metastability is not propagated to the downstream logic.

### C. *Noise Elimination from Formal Verification*

During the formal verification, the tool eliminates noise by identifying the timing exceptions that do not require formal verification based on certain criteria. The different such scenarios are described below:

1) NO PATH Exceptions: These exceptions for which:
    a. No valid startpoints/endpoints were found in the design, or they are not clocked.
    b. There is no sensitizable combinational path found between startpoint and endpoint.
2) SKIPPED Exceptions
    a. Async Clocks: These are exceptions having asynchronous launch and capture clocks.
    b. I/O Ports: These are exceptions that cannot be verified at current block level and can only be verified at a higher level where the tool is able to see the logic that drives the input ports.
    c. MCP Hold < Setup: These are MCPs having hold shift value less than setup shift value of a corresponding MCP. In such cases the hold MCP does not relax timing on the path anymore than the setup MCP already does.
    d. Duplicate: Any duplicate exceptions are skipped from formal verification.
3) WAIVED Exceptions: These corresponds to timing don't care exceptions that cannot be formally proven and should be excluded from the verification process upfront.

## II. TIMING EXCEPTION ASSERTION VERIFICATION

### A. *Proposed Methodology*

The proposed methodology for timing exception verification is depicted in Figure 2. Once the formal

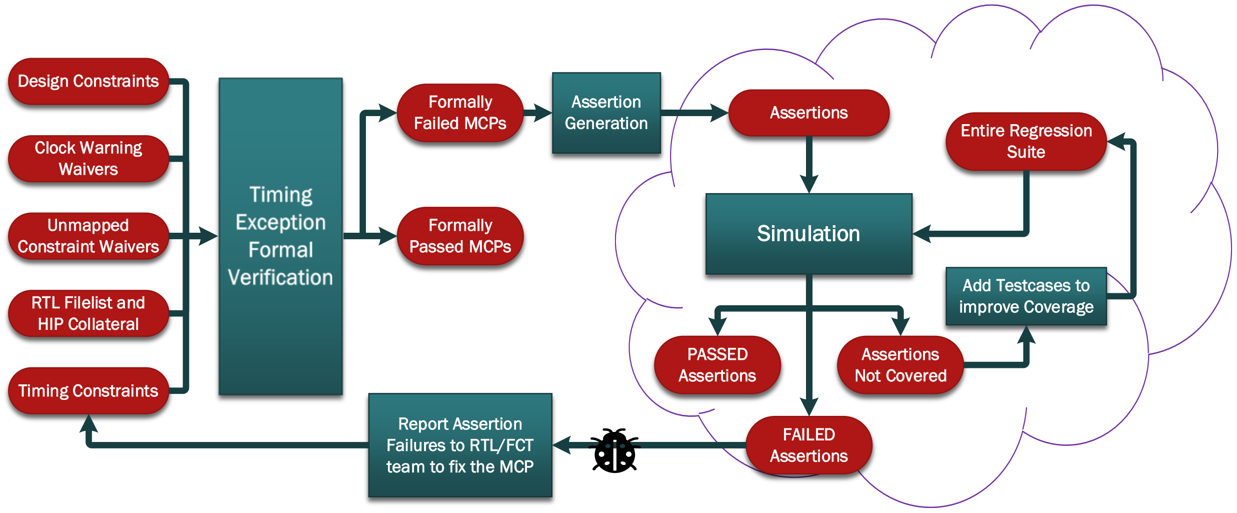

Figure 2: Timing Exception Assertion Verification Methodology

verification of timing exceptions is completed, assertions are generated for all the exceptions that has failed the formal verification. These assertions capture the functional behavior of the design that must be satisfied for the exceptions they are associated with to be correct. The assertions are then verified in functional simulation by running the entire verification regression suite of the design. The false path (FP) assertions check that the condition required to propagate a transition from startpoint to endpoint can never be true. The multi-cycle path (MCP) assertions check that when the startpoint transitions then, either in that cycle the condition required to propagate the change from the startpoint to the endpoint should not be true, or in the next cycle the endpoint should not transition. Thus, any assertion failure in functional simulation regression would represent real-world situations where the specified exception behavior does not hold. In this way, all the incorrect timing exceptions can be identified from the original timing constraints.

## *B. Plugging-in Assertions in RTL Simulations*

The standard cells used in the design have different implementations in RTL synthesis and simulation models. Since the assertions are generated from the synthesis model, if they are directly used in simulation, then it would result in cross-module reference resolution errors (XMREs). Such issues can be resolved by restricting the tool from tapping the internal signals of standard cells and refer signals only at the boundary of standard cell wrappers while generating the assertions. However, the signals inside standard cell wrappers are still tapped if there are any sequential elements present between the signal and its input ports. In such cases, an additional mapping file must be provided which maps the synthesis hierarchy to the corresponding simulation hierarchy so that the assertions are generated with proper simulation hierarchies and can be directly used in simulation.

## *C. Signing off the Uncovered Assertions*

The sign-off criteria for uncovered assertions is shown in Figure 3. The assertions that remain uncovered during functional simulation regression can be because of the following two scenarios:

1) Exception startpoint has not toggled.
   In such cases, all the untoggled exception startpoints must be reviewed and confirmed if these are indeed static signals or not. If they are, they must be constrained as such for the formal verification so that formal verification passes for all the timing paths with static starpoints and assertions are not generated for these paths. If they are not static, then proper testcases should be run that can cover the startpoints and verify the assertions.
2) Launch clock not reaching the startpoint.
   In such cases, all the uncovered assertions must be reviewed and confirmed that the launch clocks are not expected to reach the respective startpoints. Otherwise, additional testcases must be run to cover these assertions.

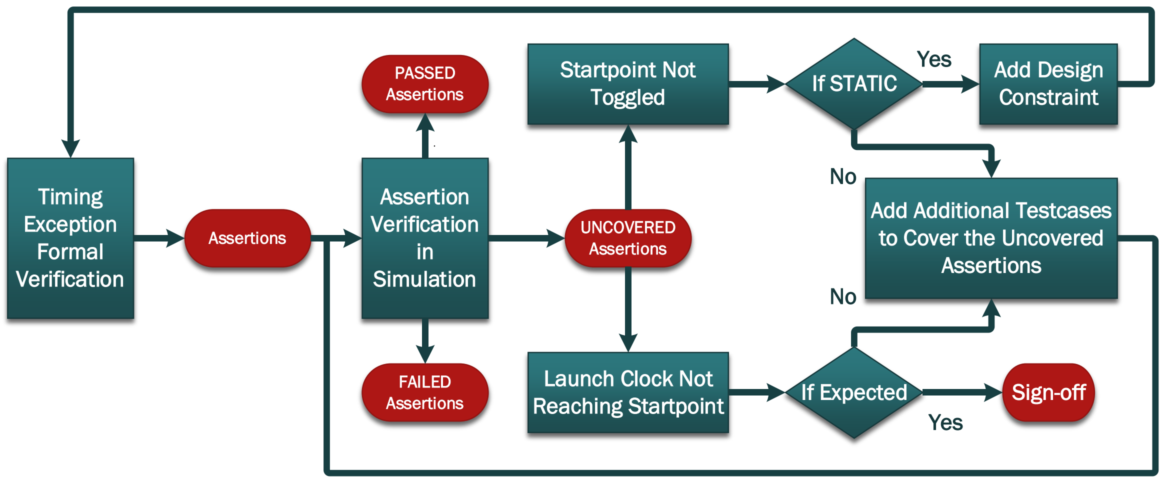
Timing Exception Formal Verification
Assertions
PASSED Assertions
Assertion Verification in Simulation
FAILED Assertions
UNCOVERED Assertions
Startpoint Not Toggled
Launch Clock Not Reaching Startpoint
If STATIC
Yes
No
Add Design Constraint
Add Additional Testcases to Cover the Uncovered Assertions
No
If Expected
Yes
Sign-off

Figure 3: Uncovered Assertions Sign-off Criteria

## IV. Results

The GFX SOC design under consideration had nearly 2M timing constraints including more than 600 master clocks, 1100 generated clocks and 20K MCP specifications. The timing constraint formal verification was done at the partition level as per the proposed methodology. The unmapped constraints and clock propagation issues reported were reviewed with FCT team and the critical ones were fixed in the timing constraints while waivers were added for the rest of the violations, thereby ensuring zero clock propagation issues and 100% timing constraints mapping across all the SOC partitions by the time of final RTL milestone release. The several unmapped constraints and clock warnings that were fixed in the timing constraints are tabulated in Table 1.

The MCP formal verification results are tabulated in Table 2. More than 5K MCPs have failed the formal verification corresponding to which a total number of 250K assertions were generated. The formally failed MCPs were separated into functional and DFT related MCPs after reviewing with DFT team. Out of the 5K formally failed MCPs, 72 were functional (non-DFT) MCPs corresponding to which there were 6602 assertions in total. These assertions were integrated into functional simulation regressions where 154 assertion failures were observed. All the assertions failures in simulation were correlated back to 7 MCPs. The simulation failures were debugged with the RTL team after generating FSBD for all the failing testcases and the incorrect MCPs were fixed in the timing constraints and were incrementally verified. Rest of the assertions have either passed or remained uncovered in simulation. For the uncovered assertions, the startpoints were reviewed with RTL team. For all the non-static startpoints, additional testcases were run and got them covered while the rest of the startpoints were reviewed to be static signals. The assertion validation results are tabulated in Table 3.

Table 1: Unmapped Constraints and Clock Warnings Fixed in Timing Constraints

| # | Issue Description | No. of Issues Fixed |
|---|---|---|
| 1 | Syntax Error in SDC Constraint | 5 |
| 2 | Net has clock-like behavior, but no clock propagates to it | 7 |
| 3 | Clock propagation was stopped by a clock | 2 |
| 4 | Generated clock has an invalid edge specification | 1 |
| 5 | There is a clock-to-clock false path, but the clocks have the same master clock and are not exclusive | 16 |
| 6 | The clock crossing from clock to clock has been specified as logically exclusive, but there is a nonexclusive crossing between these clocks | 2400 |
| 7 | Clocks have been specified as physically exclusive, but they can co-exist in the design at the same time | 2 |
| 8 | Clocks are logically exclusive, but they have not been specified as such | 17 |

Table 2: SOC MCP Formal Verification Results

| Total Number of MCPs | No. of PASSING MCPs | No. of FAILING MCPs | No. of NO PATH MCPs | No. of SKIPPED MCPs | No. of WAIVED MCPs |
|---|---|---|---|---|---|
| 21743 | 12774 | 5879 | 145 | 753 | 145 |

Table 3: SOC Functional (non-DFT) MCP Assertion Verification Results

| No. of Functional (non-DFT) MCP Formal Failures | No. of Assertions Generated corresponding to Functional (non-DFT) MCPs | No. of PASSED Assertions | No. of FAILED Assertions | No. of UNCOVERED Assertions | No. of MCPs corresponding to Assertion Failures |
|---|---|---|---|---|---|
| 72 | 6602 | 1494 | 154 | 4954 | 7 |

## V. Conclusion

The proposed methodology has provided a faster and more efficient way for timing constraint verification that can be deployed at early stages of the design development cycle. The proposed methodology has enabled to identify and resolve all the unmapped constraints and clock propagation issues as well as formally verify the timing exception by the final RTL milestone release, thereby achieving significant shift-left in the overall timing closure. All the incorrect timing exceptions were identified by assertion verification in the functional simulation before the timing closure of the design, preventing any potential silicon escapes and costly re-spins. Therefore, the proposed methodology has improved the overall accuracy and reliability of timing constraints, leading to a higher quality design. The proposed methodology is scalable, and its application can be extended beyond the GFX SOCs. Its principles and techniques can be easily applied to other complex SOC designs, leading to improved timing constraint verification efficiency and reduced time-to-market for a wide range of semiconductor products.

## References

[1] A. Khandelwal, A. Gaur and D. Mahajan, "Gate level simulations: verification flow and challenges," EDN, 5 March 2014. [Online]. Available: https://www.edn.com/gate-level-simulations-verification-flow-and-challenges/.

[2] P. Limmer, D. Moeller, M. Mueller and C. Roettgermann, "Validation of Timing Constraints on RTL: Reducing Risk and Effort on Gate-Level," in *DVCON Europe*, 2016.