The beginning of new norm: CDC/RDC constraints signoff through functional simulation

Suhas D S, Ponsankar Arumugam, Deepmala Sachan, Ritesh Jain
Intel Technology India Pvt. Ltd.
Bengaluru, Karnataka, India
{suhas.d.s| ponsankar.arumugam|deepmala.sachan|ritesh1.jain}@intel.com

As the field of Very Large-Scale Integration (VLSI) design advances, the complexity of integrated circuits continues to grow. Clock and reset domain crossings have emerged as critical challenges that demand meticulous attention during the design and verification phases. Clock Domain Crossing (CDC) and Reset Domain Crossing (RDC) refer to scenarios where signals traverse between distinct clock or reset domains, presenting potential hazards such as metastability, data corruption, and unintended behavior. CDC and RDC checks are done based on assumptions which are coded in the form of constraints and any wrong assumptions can lead to potential hazards in the design, which can turn into silicon failure. This paper explores the importance of static checks in the context of CDC / RDC verification, where architectural assumptions, such as constant signals, static signals, gray encoding, qualifiers, and reset ordering are utilized to generate the SystemVerilog Assertions (SVA). Eventually these assertions should be fully validated in functional simulations with 100% SVA coverage to ensure the correctness of design assumptions.

Keywords—RTL design; CDC/RDC constraints; Design constraints; Assertions; Simulation.

I. INTRODUCTION

Graphics SoCs are high-performance complex designs with multiple clocks and resets interacting between multiple modules having CDC/RDC scenarios. Typical Graphics SoCs contains hundreds of IP's (internal/external) and multiple subsystems having 100’s of clocks, which elevates the Clock Domain Crossing(CDC) complexity. The SoC complexity is tabulated in Table 1.The integration of various IP in a complex SoC necessitates dealing with different clock domains, each operating under its own clock signal with varying frequencies and phases. Furthermore, reset signals play a vital role in initializing and maintaining the stability of digital designs. Clock Domain Crossing (CDC) and Reset Domain Crossing (RDC) checks using industry standard tool ensure the reliable operation of these interconnections. This paper takes you to the journey of how the design assumptions used in CDC/RDC static checks can be further scrutinized and fully validated in functional simulation before RTL signoff to avoid any silicon surprises.

Table 1: SoC complexity

Clocks	190 +
Clock Domains	60+
Resets	50+
Reset Scenario	39+

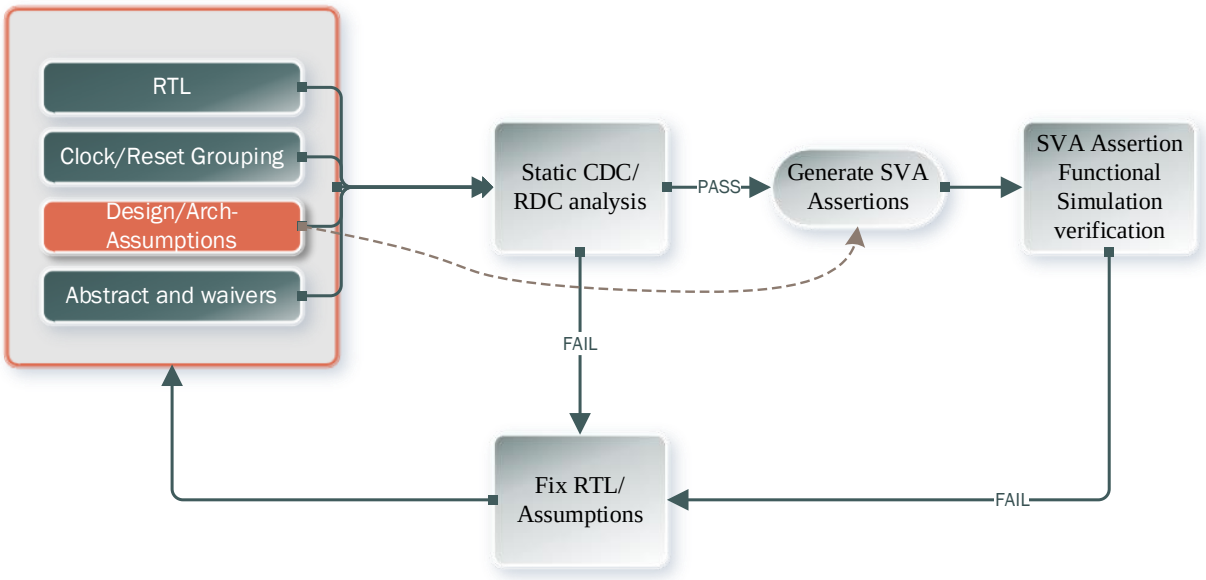

Figure -1: Flow diagram of SVA based functional simulation signoff for CDC/RDC Constraints.

II. Clock Domain Crossing (CDC) and Reset Domain Crossing (RDC)

A. CDC - Synchronizing Data Across Clock Domains

CDC involves the transfer of data between different clock domains within an SoC. Asynchronous clock domains can lead to metastability issues, which can result in data corruption. To mitigate these challenges, CDC verification tools check for violations and recommend synchronization techniques like two-flop synchronizers, gray coding, and appropriate qualifiers to ensure reliable data transfer.

B. RDC - Managing Reset Signals Across Domains

RDC addresses the synchronization of asynchronous reset signals. Inconsistent or improper reset signal handling can lead to race conditions and functional anomalies. Static analysis check for RDC verify the proper ordering and propagation of reset signals between domains. Architectural assumptions about reset ordering and signal constants guide this analysis.

III. Architectural Assumptions during Static Analysis

Static analysis checks are indispensable in verifying CDC and RDC in complex SoCs. These checks rely on various architectural assumptions to guide their analysis. We have used following design assumptions for our CDC analysis.

1. **Constant Signals:** Constant signals are assumed to remain stable and unchanged within their respective domains. The SVA's validate these assumptions by checking for any potential changes or glitches during signal crossing.
2. **Quasi-Static Signals:** Quasi-Static signals are considered as signals that remain at a constant logic level within their respective domains. The SVA's assess the stability of these signals during CDC and RDC analysis.
3. **Gray Encoding:** Gray encoding is used to encode data for safer transitions between clock domains. The SVA's ensure that gray-encoded data maintains its integrity during crossing.
4. **Qualifiers:** Qualifiers are used to indicate specific conditions for data transfer between domains. Assumptions about qualifier behavior are checked to verify that data is properly synchronized.
5. **Reset Ordering:** The ordering of reset signals is crucial in maintaining system stability. The SVA's validate reset ordering assumptions to prevent race conditions.

These design assumptions play pivotal role in Full chip CDC analysis; hence it is imperative to validate them before CDC closure. One of the steps taken to validate these assumptions is to generate the SystemVerilog Assertions (SVA's) corresponding to each design assumption and later validate them in functional simulations.

IV. Methodology recommendation for SVA based Verification

While static analysis tools provide critical role in CDC/RDC analysis, functional verification through simulations is equally necessary to validate the correctness of architectural assumptions made during static analysis. Simulations mimic real-world scenarios which help to ensure that the designed system behaves as expected under different conditions and edge cases. The fundamental goal of this paper is to provide holistic CDC/RDC constraints for signoff methodology by validating the constraints using SystemVerilog Assertions in functional simulation.

A. Generate the SystemVerilog Assertion (SVA) for each architecture assumptions.

1. **SVA for Constant signals:** single property for WRONG-VALUE conditions have been coded to validate Set Case Analysis (SCA).
2. **SVA for Quasi-static signals:** Three properties have been coded to validate such signals.
   NO-TOGGLE - To check if the quasi-static signal has not toggled during simulation.
   SETUP-TOGGLE - To check if toggling of the signal is causing setup violation.
   HOLD-TOGGLE - To check if toggling of the signal is causing hold violation.
3. **SVA for Gray encoded signals:** single property for DETECT-GRAY has been coded to identify the data bus changes, violating gray-encoding protocol.
4. **SVA for Reset order sequence:**
   a. *Reset-filter-path for RDC:* Three properties have been coded to validate such crossings.
   DETECT-RFP-SEQUENCE- To check if "to-reset" is already asserted whenever "from-reset" asserts.
   STATIC-CONDITION – To checks if the resets have not toggled during the simulation and if the from-reset is asserted and to-reset is de-asserted.
   DETECT-RPF – To check if to-reset is asserted for at least margin amount of time before from-reset asserts.
   b. *Reset-filter-path for async crossing* single *property:* "SRC-RST-DES-CLK" has been coded to checks whether the destination-clock is gated before the source-reset is asserted.
5. **SVA for Qualifier:** Single property "DATAHOLD-SOFT" has been coded to check if source data is held stable when the enable(qualifier) signal is high.

B. Plugin the SV assertions in functional simulation and Validation of assertions.

All SV assertions corresponding to each constraint should plugged in to simulations and appropriate testcases need to be identified to exercise them either at SoC or subsystems. The goal is to maximize the assertion coverage. Non-covered assertions require thorough analysis followed by the addition of specific tests or downgrading the assertions due to architecture limitations. However, failed assertions require appropriate actions in terms of constraints correction or design fix.

C. Sign-off strategy for Uncovered Assertions.

The goal is to get 100% assertion coverage however there would be certain exceptions due to SVA coding and testbench limitations. Such properties should be thoroughly reviewed with the architect. In our Graphics SoC, we have reviewed 3.6% (300+) with architect to sign-off the uncovered assertions.

V. Detailed case studies based on the analysis of CDC SVA failures

Detailed methodology of simulation is depicted in the picture below. With growing complexity of SoC's, the simulation environment has various models. The models consist of subsystem testbench's, which depending upon use cases may include black boxed and full multi-die SoC models. Care needs to be taken to divide the entire CDC assertions across these models and make sure that proper regression/use-case and system scenarios are run to trigger all the conditions to validate the assertions. Coverage of assertions can be achieved by one scenario, but the verification engineer should make sure that all the paths to trigger the assertion have been simulated. The cohesion between designer and verification engineer plays a vital role in achieving this.

Verification engineers should be cautious and should enable metastability simulations, make sure that there are no unwanted forces, avoid test cases which rely on stimulus to be forced to a given state by the test bench, and be cautious in using power aware simulations as this can give unwanted failures or coverage. While using power aware simulations, make sure to enable the assertions which exercise power sequencing scenarios.

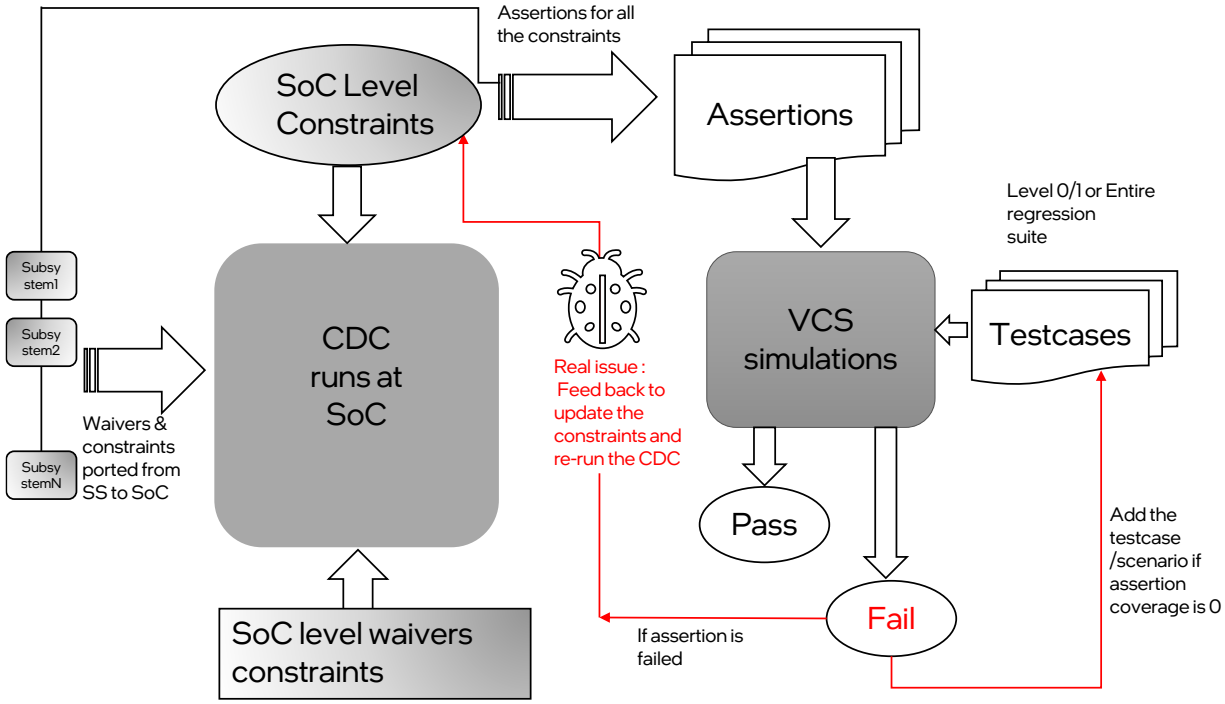

Figure-2: CDC constraints signoff flow based on SVA based validation.

A. ***Case Study 1: - False test failures due to testbench Architecture:*** Testcases are run on various TB models based on use cases where certain portion of design is stubbed out. When a design is stubbed out, a passive value is driven on input signals to avoid "x" propagation. In such cases, if value driven/forced is incorrect, the assertion will fail. In this case the testbench should be modeled to the architectural behavior. Many such instances were corrected in testbench.

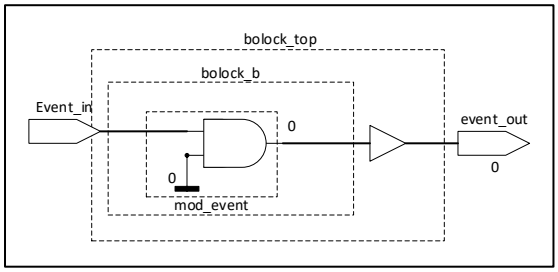

Figure -3: Schematic for false testcase failure

In the above scenario the one pin of the AND gate is tied to zero in rtl. So SCA is given w.r.t the RTL tie off value, but during simulation there is force on this signal and the testcase used to toggling from 0 -> x -> 0 which resulted in CDC SVA failure.

B. ***Case study 2: - Removal of wrongly added set case analysis:*** These constraints were added on the output port of the unit level runs to match the abstract validation errors at SoC. This need not to be set, as internal logic will set according to the mode of operation.

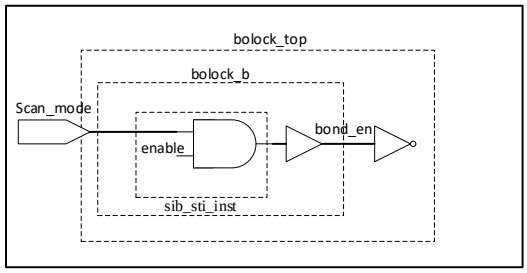

Figure -4: Schematic for wrong set case assumption

We can see that the “enable” is ANDed with "scan_mode" inside the “sib_sti_inst” module, and in “scan_mode” that will be "0" in functional test. So, in this case the set case value for “bond_en” should be 0 for functional model analysis.

VI. RESULTS AND CONCLUSION

As part of this methodology, we recommend having 100% assertion coverage for constraints signoff. However certain exceptions can be taken due to specific SVA coding and testbench limitation. Such properties must be thoroughly reviewed to get full confidence.

In our multi Die Graphics SoC 96.7% SVA coverage was achieved with the remaining 3.3 % properties are categorized as invalid due to architectural assumptions.

During our journey of validating assumptions through SVA, we came across many challenges and learning. Some of key validation learnings are:

1. **Issues with reset initialization:** All RDC assertions with respect to reset domain crossing require analysis from time zero as it checks for destination reset to be asserted way before source reset.
2. **Strategy to preload or initialize memories with in-active values:** In certain scenarios this approach has been used to avoid false failures and uncovered assertions.
3. **Failures due to power off conditions:** some false assertion failures have been noticed in power aware simulations due to “x” propagation during the power off. The assertions need to be modeled accordingly to disable them during power off condition.
4. **Assertions having in active clocks in functional mode:** since our checks are done in functional mode, Test/DFX clocks and logic are not active, hence the assertion containing them can't be hit as they are not active. The constraints were reviewed and SVA’s were removed for such instances.
5. **Absence of abstract for HIP/Memories:** All hard IP and memories should have an abstract model to avoid unnecessary assertions for simulation overhead.
6. **Testbench Modelling:** To avoid false failure, testbench models should be chosen appropriately so that undriven or falsely driven signal doesn’t give unwanted coverage or failure.
7. **Exploration of formal for assertion validation:** The Graphics SoC is multi die with large gate count and long simulation times. Our next process is to implement formal tools to achieve shorter runtime and quicker turn around. The challenge will be to have appropriate assumptions.

Along with the modification of testbench to address uncovered assertions, we have made following constraint updates to strengthen the overall CDC/RDC analysis.

1. **Removal of wrongly added “reset-filter-path”:** It was wrongly added during CDC analysis to reduce huge waiver count: this was put to waive only where to_obj is present, but assertions are generated for from_rst to to_rst, which is huge verification overhead.
    a. reset_filter_path -from_rst “rstA” -to_rst “rstB” -to_obj “flop/Q” -type rdc
2. **Correction of SCA constraints:** Few wrongly added constraint were corrected to its functional values.

3. **Correction on data qualifier:** All async FIFOs require proper qualifier constraints to avoid false assertions.
    a. qualifier -dest_qual <FIFO-empty-signal> -from_obj <your-FIFO-register > -to_clk <destination-clock>
    b. qualifier -src_stable -dest_qual <read-pointer> -from_obj <your-FIFO-register> -to_clk <destination-clock>
4. **Consideration of Data Hold assertions:** For all memory or FIFO assertions, consider every data bit which are hard to hit. Hence first and last data bit of memory/FIFO were considered to reduce overall assertion count which has drastic impact on simulation TAT.
5. **Removal or correction of wrongly added set case analysis:** These constraints were added on the output port of the unit level runs to match the abstract validation errors at SoC. This need not to be set, as internal logic will set according to the mode of operation.
    a. set_case_analysis -name "soc.par1.scan_or_out" -value 0

While we are exploring formal as a way to accelerate the overall execution , we strongly recommend SVA based constraint verification to go hand in hand with standard CDC execution cycle before RTL signoff to avoid any silicon failures.

REFERENCES

[1]. Amit Kulkarni, Suhas D S, Deepmala Sachan, “Evolution of CDC recipe: Learning through real case studies and methodology improvements”, in DVCON 2021

[2]. Rohit Kumar Sinha, “Enhancing Quality and Coverage of CDC Closure in Intel’s SoC Design”, in DVCON 2020