Accelerating CDC Verification Closure on Gate-Level Designs

Anwesha Choudhury, Ashish Hari
anwesha_choudhary@mentor.com, ashish_hari@mentor.com
Design Verification Technologies
Mentor Graphics

Abstract: **Increasing usage of multi-clocking architecture to meet high performance and low power requirements of the modern SOCs makes clock domain crossing (CDC) verification a critical step in design verification cycle. CDC verification is not only necessary on RTL; it is also a necessity on gate-level designs due to the possibility of the introduction of CDC errors during the synthesis phase that can lead to silicon failure. However, CDC closure on gate-level designs continues to be one of the most difficult tasks during the verification cycle because of very high setup effort, stretched performance and high noise in the results. In this paper, we review the root cause of these challenges and introduce an automated approach to overcome these difficulties. The proposed methodology is based on learnings from complete verification process and utilization of knowledge gained from prior verification steps to automate and accelerate gate-level CDC verification closure. The proposed methodology enhances gate-level CDC verification experience for the user by automating the setup, accelerating performance, eliminating noise, and easing debug.**

I. INTRODUCTION

CDC verification ensures that signals pass across asynchronous clock domains without being missed or causing metastability. Traditionally, CDC verification is done on a register-transfer level (RTL) representation of the design. However, during the synthesis stage, when the design is transformed from RTL to gate level, various new issues can be introduced that may eventually lead to chip failures. So, even after CDC verification closure at the RTL, it is important to perform CDC verification on a gate-level design to detect and address new issues.

Changes introduced in the netlist during synthesis, such as timing optimization of synchronization logic and insertion of design-for-test (DFT) and low-power circuitry may cause incorrect behavior in CDC synchronizers, introduce new CDC paths or break valid CDC paths. For example, new clock domain crossings can be introduced due to insertion of low-power logic as shown in *Figure 1*. Similarly, due to scan logic insertion, a clock tree can be impacted if the correct mode design constraints are not specified, as shown in *Figure 2*.

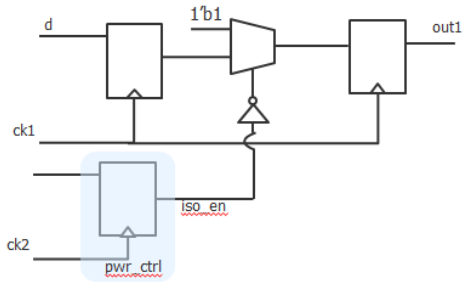

Figure 1: CDC path impacted by power logic

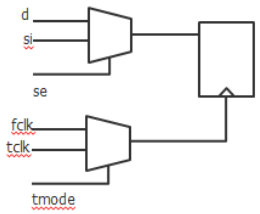

Figure 2: Clock and data path impacted by DFT logic

Additionally, faulty implementation of combinational logic by synthesis tools may result in glitches on control and data paths. As shown in as shown in *Figure 3*, a valid mux-based synchronizer is converted to a combinational logic which is logically correct, but can propagate a glitch from asynchronous transmit domain causing chip failure.

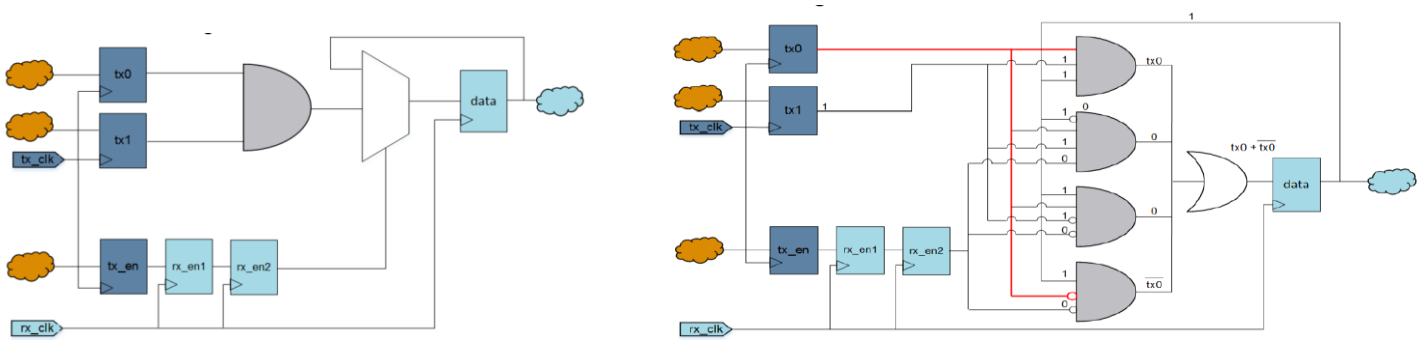

Figure 3: Mux logic converted to combinational logic with glitch

In the subsequent sections, we explore the challenges and redundancy associated with the use of the traditional CDC verification flow for gate-level designs. We propose a new approach that leverages the RTL CDC run design constraints, automatically infers scan and enable logic, understands transformations during synthesis and utilizes this knowledge during CDC analysis. The proposed approach also reuses CDC run waivers from the RTL run at the gate level by appropriate transformations and performs data mining on results, thereby accelerating CDC verification closure on gate-level designs.

II. Traditional CDC Verification Flow

An overview of the traditional CDC verification flow is shown in *Figure 4*.

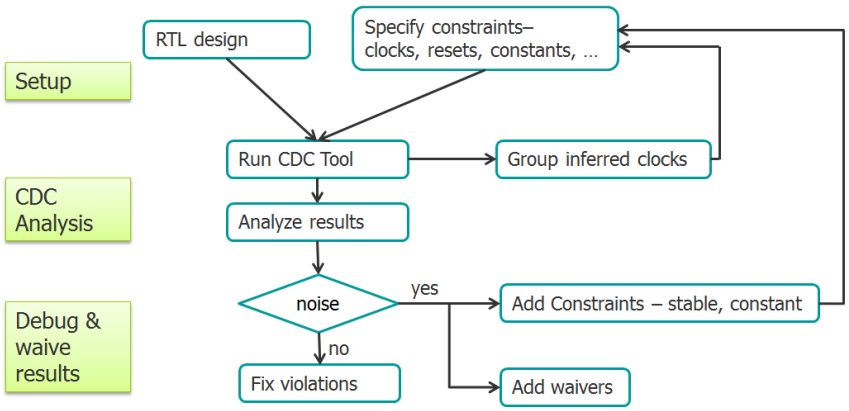

Figure 4: CDC verification flow on RTL

Primarily, CDC verification can be divided into three broad stages:

- Setup: This is a critical component of the CDC verification flow. The user needs to specify appropriate constraints for the design to obtain accurate CDC results. Missing or wrong constraints can lead to false or missing violations.
- CDC analysis: At this stage, the CDC verification tool analyzes the design using the user-specified constraints and reports the CDC results.
- Debug: At this stage, the user reviews and debugs the results reported after CDC analysis. If there are too many false violations, the user can update the constraints and rerun the analysis. Finally, for each violation, the user performs one of the following:
    - Fix issues and add status notes or comments to explain the fixes or the analysis. If added, the status or the comments are expected to be visible during all subsequent CDC runs on the same design, to eliminate the need for reanalysis of the same violation.

- Waive the expected violations. These violations may be expected and harmless. These could be the result of a quasi-static transmitting signal, a path that is intended to be reviewed later, or simply a path qualified by the designer. The expectation is to hide the waived violations in the subsequent CDC runs as they are already analyzed and qualified.

After the completion of the above stages along with the CDC-clean RTL design, a constraints setup and a waiver setup is available for use in all the subsequent CDC analysis on the design.

III. Why Does Traditional CDC Verification Flow Stumble At Gate Level

The most common reasons that make gate-level CDC verification closure a daunting task are as follows:

- Significant setup effort:
  - All designs have refined CDC design constraints setup at the RTL with which the design is verified to be CDC clean. These RTL constraints cannot be reused during CDC verification at the gate level because of changes in the module and signal names post synthesis and technology mapping. All design constraints that refer to RTL signals and modules no longer work. Manual conversion of constraints is not feasible as it involves high effort and needs knowledge of name transformations done by the synthesis tool. Writing new setup constraints is not desirable as it requires several iterations to fine-tune the constraints and thus leads to significant delay in the verification schedule.
- Glitches can be introduced on CDC paths that were safe at the RTL:
  - A common situation is when a mux in the data path is transformed as combinational logic that can possibly glitch. This situation requires special approach to identify and fix glitch scenarios.
- The design bus signals are split into multiple single-bit signals at the gate level, causing several issues:
  - Differentiation between data and control signal is lost and traditional CDC verification tools treat every path as a single-bit control path.
  - The CDC crossing count increases many times, resulting in noise and a higher debug effort.
  - Design size increases many times as compared to the RTL, stretching the capacity limits of CDC verification tools.
- High debug effort due to increase in violations and noise:
  - A user adds waivers during CDC analysis at the RTL for acceptable scenarios. However, waivers written on RTL results do not work at the gate-level due to signal name and topology changes. This makes CDC closure more difficult as now the verification engineer needs to reanalyze and waive CDC paths that are already qualified at the RTL. This leads to undesired redundancy in the verification process.
  - False violations can increase due to the interference of scan logic. User intervention is needed to correctly specify the functional mode.

IV. Proposed Gate-Level CDC Verification Flow

In this paper, we propose a systematic, automated approach that addresses the limitations of the traditional CDC verification for gate-level designs and significantly reduces the verification effort.

Figure 5 illustrates a high-level CDC verification flow on RTL to netlist and how to enhance the flow to reuse knowledge about setup and waivers at the RTL for downstream gate-level CDC verification. Typically, CDC closure is one of the necessary RTL sign-off criteria. So every design at RTL stage would have gone through CDC verification. After the RTL finalization, a design goes through synthesis and a netlist is generated. As mentioned in the previous sections, a fresh CDC run on the netlist will require huge setup effort and will lead to excess noise in results. To address this issue, in this methodology we propose to reuse the constraints and waivers that were used during CDC verification at the RTL.

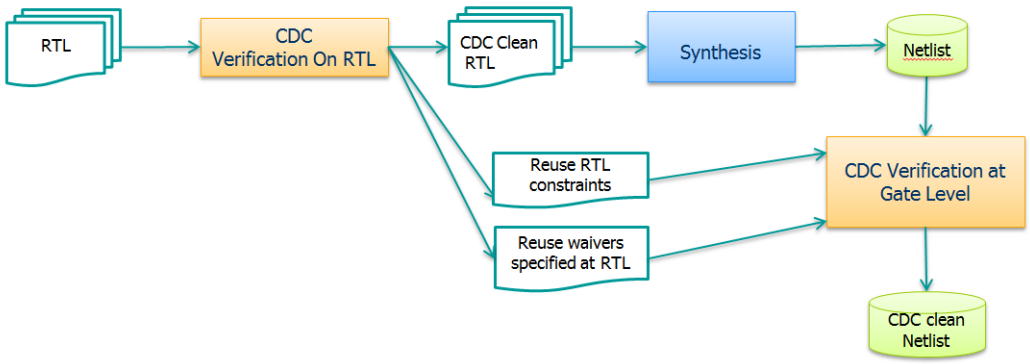

Figure 5: CDC verification flow from RTL to netlist

The proposed gate-level CDC flow is illustrated in *Figure 6*. The flow works on a gate-level netlist along with RTL constraints or gate-level constraints or SDC for the gate-level design. We propose a setup step in which, the tool infers the design mode constraints and detects the scan and test signals. Then, after the user qualifies the setup, the actual CDC analysis on netlist happens.

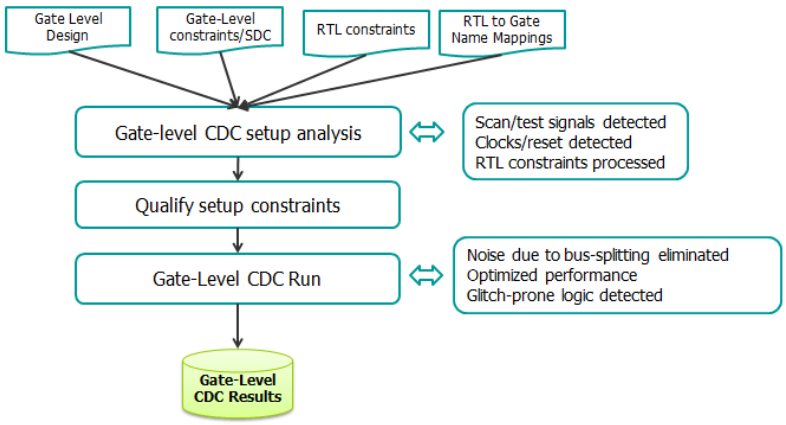

Figure 6: Proposed Gate-level CDC Verification Methodology

The sections below describe the key components of the proposed solution that enable it to handle the challenges of the conventional approach.

A. Setup Automation

The proposed approach saves user effort and iterations required to create the design and tool constraints for the gate-level CDC run. Users just need to provide the following information that should be readily available from previous steps in the design development and verification flow:

- The constraints used for CDC run at the RTL can be provided as is. The proposed solution can automatically transform these to match the gate-level signal and module names. The methodology understands the naming transformation during synthesis and utilizes this knowledge to convert signal and module names from RTL format to gate-level format. This automation saves user effort in creating and refining the setup through multiple iterations and also ensures reduced false noise as the constraints are already validated in RTL CDC analysis. An example of this automation is illustrated in *Figure* 7. The custom synchronizer and quasi-static signal specifications in the RTL are automatically converted to match the gate-level design signal and module names.

RTL constraints	Auto-generated Gate-level constraint
cdc custom sync cs_two_dff	cdc custom sync cs_two_dff_0_1 cdc custom sync cs_two_dff_1_1
cdc signal fifo_inst.bus_read[0] -stable	cdc signal fifo_inst.bus_read.rtlc_reg_0.iq -stable

Figure 7: Automatic conversion of RTL constraints to gate-level constraints

- The accuracy in reuse of RTL constraints is further improved if RTL-to-gate name mappings are provided to this methodology. Name-mappings are generally available in a file after logic equivalence check (LEC) is run on the synthesized design. The following is a snapshot of the mapping file generated by the FormalPro LEC tool :

```
D Flip-Flops A
---------------

Name:     \A.u_CONF_REQ.ck_reg_data_r_reg(0)  \B.u_CONF_REQ.rtlcreg_ck_reg_data_r_0
Name:     \A.CORE_M0.u_CONF_REQ.ck_reg_data_r_reg(1)  \B.CORE_M0.u_CONF_REQ.rtlcreg_ck_reg_data_r_1
Name:     \A.CORE_M0.u_CONF_REQ.ck_reg_data_r_reg(2)  \B.CORE_M0.u_CONF_REQ.rtlcreg_ck_reg_data_r_2
Name:     \A.CORE_M0.u_CONF_REQ.ck_reg_data_r_reg(3)  \B.CORE_M0.u_CONF_REQ.rtlcreg_ck_reg_data_r_3
Name:     \A.CORE_M0.u_CONF_REQ.ck_reg_data_r_reg(4)  \B.CORE_M0.u_CONF_REQ.rtlcreg_ck_reg_data_r_4
Name:     \A.CORE_M0.u_CONF_REQ.ck_reg_data_r_reg(5)  \B.CORE_M0.u_CONF_REQ.rtlcreg_ck_reg_data_r_5
Name:     \A.CORE_M0.u_CONF_REQ.ck_reg_data_r_reg(6)  \B.CORE_M0.u_CONF_REQ.rtlcreg_ck_reg_data_r_6
Name:     \A.CORE_M0.u_CONF_REQ.ck_reg_data_r_reg(7)  \B.CORE_M0.u_CONF_REQ.rtlcreg_ck_reg_data_r_7
Name:     \A.CORE_M0.u_CONF_REQ.ck_reg_data_r_reg(8)  \B.CORE_M0.u_CONF_REQ.rtlcreg_ck_reg_data_r_8
```

Figure 8: Snippet of the name mapping file

  This readily available LEC output file can be provided during CDC analysis on a gate-level design. The proposed approach utilizes name mapping information from the given files to understand how the signal names change from RTL to netlist. This knowledge is utilized during the automatic conversion of RTL constraints to gate-level constraints. This information also enables the correlation between RTL and gate-level results, so that the user can identify new CDC issues introduced during synthesis.

- Users can also provide the gate-level SDC file. The proposed methodology automatically extracts CDC information from the given SDC and uses it for CDC analysis.

The next level of automation is required to infer constraints for the logic that was not present in the RTL but got inserted during synthesis and later stages. For example, DFT logic is inserted later in the flow. However, it is important to verify the CDC design in the right mode, that is, the functional mode. If the tool does not differentiate between test and functional modes, it leads to more gated clocks and hence extra noise and false crossings. In the proposed methodology, such mode and constant settings are inferred by analyzing the design topology using software algorithms on the design graph. This ensures that the user need not manually analyze false crossings and then constraint the design in the appropriate mode. With the suggested approach, the user just runs the CDC anlaysis and reviews the constraint guidance provided by the tool.

An example of the scan logic impacting CDC crossings is shown in *Fig*ure *9a*. If mode constraints are not specified, then with traditional CDC verification approach, the CDC path will be reported as unsynchronized crossing. However, in the proposed methodology such logic is analyzed to detect scan enable and test mode signals and correct constants are set on them automatically to ensure the design operates in functional mode and synchronizers are detected as shown in *Fig*ure *9b*. This approach eliminates false noise due to insertion of extra logic, also saves user effort in specifying the constraints manually.

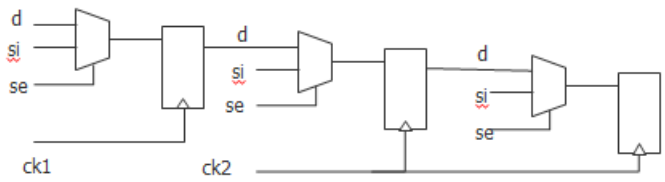

Figure 9a: DFT inserted CDC path

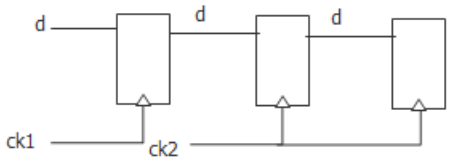

Figure 9b: CDC path with synchronizer in functional mode

This automated setup approach significantly reduces the effort and iterations required in creating constraints for gate-level CDC analysis. It also ensures consistency and reduces dependency on verification engineer's judgment by automating the process. The generated setup is available for final review and qualification by the verification engineer.

B. Elimination of Noise in CDC Analysis

Honoring RTL constraints and inferring functional mode constants ensures that the setup is correct and minimal noise is reported during CDC analysis on the gate-level netlist. However, there would still be considerable noise due to splitting of vector signals during synthesis as this leads to huge increase in the number of crossings. The proposed methodology understands the splitting during synthesis and utilizes this knowledge during CDC analysis to correctly identify the control and data crossings. The identification of data crossings drastically reduces the noise and also brings gate-level results closer to RTL results. In addition, the RTL signal name is reported so that users can correlate gate-level results with RTL results. An example of this approach is illustrated in *Figure 10*. A 16-bit data crossing in the RTL gets converted to 16 individual single-bit crossings in the netlist. In the proposed approach, a single data crossing and its correlation to the RTL crossing is reported.

RTL Data Crossing	Gate-Level Crossings (Traditional verification methodology)	Gate-Level Crossings (Proposed verification methodology)
tx_clk: tx_inst.bus_read[15:0] rx_clk: sync_inst.bus_read[15:0]	Crossing 1 : tx_clk: tx_inst.rtlc_reg_bus_reg_0.iq rx_clk: sync_inst.rtlc_reg_bus_reg_0.iq Crossing 2 : tx_clk: tx_inst.rtlc_reg_bus_reg_1.iq rx_clk : sync_inst.rtlc_reg_bus_reg_1.iq ... Crossing 15 : tx_clk: tx_inst.rtlc_reg_bus_reg_15.iq rx_clk: sync_inst.rtlc_reg_bus_reg_15.iq	tx_clk : tx_inst.rtlc_reg_bus_reg_0.iq (RTL signal : tx_inst.bus_read[15:0]) rx_clk : sync_inst.rtlc_reg_bus_reg_0.iq (RTL signal : tx_inst.bus_read[15:0])

Figure 10: Data crossing identification

This automation reduces the violation count and guides the user to differentiate between control and data path crossings, and to correlate the results with RTL results.

C. Reuse of RTL Waivers

The suggested methodology enables reuse of the RTL CDC run waivers. This implies that the crossings qualified by users as stable or false during RTL analysis need not be reanalyzed at the gate level. The RTL waivers are automatically transformed and applied to gate-level results. Also the status or comment attached with the crossings is automatically applied to corresponding gate-level crossings. This ensures that when the user gets the gate-level results for the first time, there is minimal noise and only new issues identified at the gate-level are highlighted.

V. CASE STUDY

The proposed methodology and the traditional CDC verification methodology were executed on a set of different customer designs with 1-100 million gates. For each design, constraints at RTL were available along with the netlist. The CDC results, runtime and memory consumption were captured for each run. In *Figure 11*, we have presented the comparison of these data for both the methodologies.

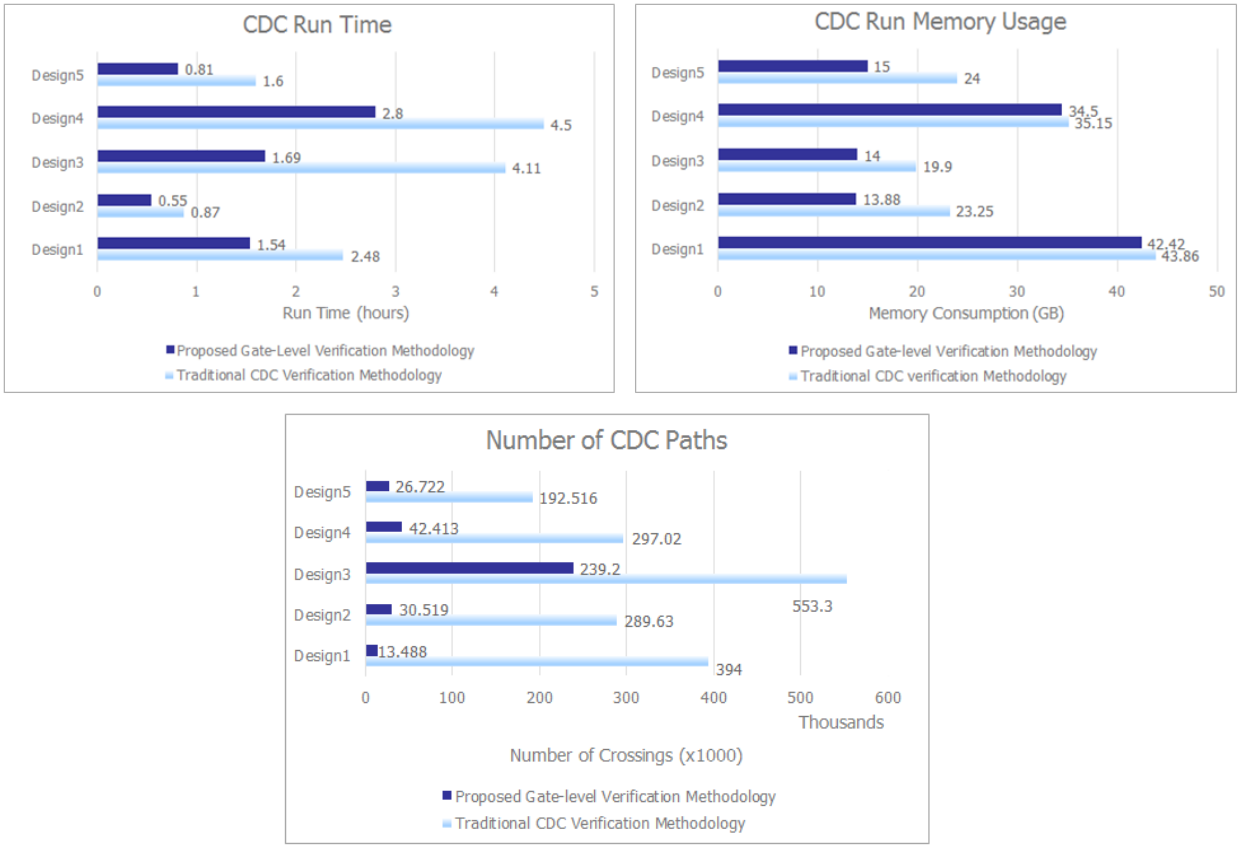

Figure 11: Comparison between proposed and traditional methodology

As evident from the data, significant improvement was observed in runtime, memory and noise with the proposed methodology. When the traditional approach was tried on these designs, it resulted in a large number of CDC paths. This was primarily due to inability to reuse the RTL constraints and waivers, missing constraints for the scan logic and splitting of vector signals. With the proposed approach the RTL constraints were applied, scan logic constraints were automatically inferred and user RTL waivers were also honored. In addition, the impact of splitting of vector signals was diminished by intelligent identification of data crossings. This approach reduced noise significantly and led to an average of 50 percent improvement in runtime and 15 percent improvement in memory consumption. As the new methodology reused RTL constraints, it required minimal setup refinements. This significantly reduced the turnaround time and effort involved in CDC verification closure on netlist.

We extended the experiments by running CDC analysis on both RTL and gate-level version of the same testcase with same set of RTL constraints. The gate-level run was executed using name-map file from LEC tool and the RTL constraints. In *Table 1*, we have presented the number of CDC paths in the RTL run and gate-level run using traditional and proposed CDC verification methodology.

TABLE I
CDC RESULTS ON RTL AND GATE-LEVEL REPRESENTATION OF SAME TESTCASE

	RTL Results	Gate-level Results (Proposed Methodology)	Gate-level Results (Traditional Methodology)
Synchronized single-bit crossings	151	150	0
Synchronized multi-bit crossings	78	77	0
Unsynchronized waived crossings	10	5	0
Unsynchronized crossings	33	1201	24881

CDC analysis using traditional CDC verification approach couldn't identify the synchronizers due to interference of scan logic in data paths. In the proposed approach, scan logic was identified and constrained automatically leading to identification of synchronizers correctly. CDC results on gate-level with proposed approach were much closer to RTL results. This ensures user can easily compare the gate-level results with RTL results and identify the issues that are introduced during synthesis.

VI. CONCLUSION

CDC analysis at gate-level is necessary; else CDC bugs introduced during synthesis can escape leading to chip failure. In this paper, we proposed a novel approach that overcomes the challenges associated with CDC verification on gate-level designs and illustrated the benefits of the proposed approach on a set of customer test cases. The suggested methodology is easy to adapt as it utilizes knowledge from the design and verification processes that occur before this stage. Automation at various stages enables the reuse of RTL constraints and waivers and seamless detection of design mode constraints. The proposed methodology enhances the CDC verification experience by automating setup, reducing noise and easing debug and thus accelerates the CDC verification closure at the gate-level.

VII. REFERENCES

[1] Clifford E. Cummings, "Clock Domain Crossing (CDC) Design & Verification Techniques Using SystemVerilog", SNUG-2008

[2] Vishnu C Vimjam and Al Joseph, "Challenges in Verification of Clock Domain Crossings", DAC knowledge center Article

[3] Ping Yeung, "*Five Steps to Quality CDC Verification"*, Mentor Graphics, Advanced Verification White Paper

[4] M. Litterick, "Full Flow Clock Domain Crossing – From Source to Si", DVCON 2016