Comprehensive IP to SoC Clock Domain Crossing Verification Using Hybrid Data Model

Anwesha Choudhury, Ashish Hari
anwesha_choudhary@mentor.com, ashish_hari@mentor.com
IC Verification Solutions
Mentor, A Siemens Business

Abstract: Clock domain crossing (CDC) verification is a critical step in the design verification cycle. Traditionally flat CDC verification on SoCs has been the preferred methodology. However as the industry inches towards multi-billion gate designs, flat CDC analysis on an SoC becomes performance intensive, time-consuming and difficult to debug, stretching verification schedules. In addition, with the increasing frequency of third-party IP usage and design reuse, clean integration gains focus along with verification of the internals of the block. So the need of the hour is a CDC verification approach that supports an IP-based flow using which SoC teams can seamlessly integrate IP and verify CDC issues that may arise due to incorrect integration. This paper presents an automated hierarchical CDC verification methodology that is based on a Hybrid Data Model (HDM), which is equivalent to a CDC IP. The model encapsulates the CDC intent of the block for accurate results and intuitive debug. It also captures the IP configurations and associated integration rules to guarantee IP designer intent is propagated and verified at the SoC level. The HDM-based CDC IP can be distributed along with the design IP and reused during SoC integration. This paper proposes an automated methodology based on HDM that leads to foolproof IP-to-SoC CDC verification and accelerates CDC verification closure.

I. INTRODUCTION

Rapid growth in design functionality, speed, size and number of asynchronous clocks poses multiple challenges for Clock Domain Crossing (CDC) verification at the SoC level. Flat CDC runs on an SoC are often performance intensive, time-consuming, result in high noise and prolongs the verification effort. There is also redundancy in debug as the same CDC bug could be replicated across multiple instances of the same module in the full chip. Also, as different modules of an SoC are developed by different designers in different geographies, there is a need for a distributed CDC verification mechanism, where each module can be verified separately and then integrated for complete CDC verification on the SoC. However this distributed CDC verification methodology should be scalable to handle multi-billion gate designs and at the same time should not compromise accuracy or ease-of-debug.

With the increasing frequency of third-party IP and design reuse, the need to accordingly adapt the traditional, flat CDC verification approach has arisen. Often the IP blocks are already verified to be free of CDC issues (i.e., they are *CDC clean*), so the focus shifts to validating the integration of IPs rather than verifying the internals of the IP blocks. Also there needs to be some mechanism to hide the IP internal details and still verify the integration comprehensively. So the need of the hour is a CDC IP-based flow that SoC teams can use to simply integrate the CDC IP models and verify any CDC issues that may arise due to incorrect integration.

In this paper we present a new hierarchical CDC verification approach that supports an IP-based flow and can be used as an alternate to the flat CDC verification flow. We describe the backbone of this solution, which is the Hybrid Data Model (HDM). HDM is equivalent to a portable CDC IP that captures the CDC intent of any block along with its integration rules. It is a generic data model that can be seamlessly reused across releases and across designs wherever the IP is reused. We also review the existing hierarchical verification methodologies, why they fail to meet the industry needs, and how an HDM-based flow addresses those challenges.

II. NEED FOR AN IP-BASED CDC VERIFICATION FLOW

Typically an SoC comprises different blocks developed by different teams spread across different geographies. CDC verification of each block is done independently as part of block-level verification signoff. Once the block is finalized and verified, it can be used in one or more SoCs. In most cases CDC verification on the blocks and on the SoC starts in parallel and eventually CDC clean blocks are integrated in the SoC. Hence, during block integration in the SoC, CDC verification for the block internals is not desirable – as it is not only redundant verification, but it also increases runtime, stretches capacity limits and eventually leads to delays in verification sign-off. So flat CDC verification is not suitable for large SoCs comprising multiple IP blocks.

The desirable use-model is where the SoC verification engineer can simply integrate the blocks and focus on block integration and top-level issues. Figure 1 illustrates the possible CDC issues in the SoC. The intra-block violations are the ones where both transmitting and receiving signals are inside the block. Such violations are detected and cleaned up during block-level CDC verification. The SoC-level verification should ensure all inter-block and top-level issues are addressed, as shown in the figure. So there is a need for a CDC verification methodology that allows independent block-level and SoC-level verification and ensures that all possible CDC issues due to incorrect integration are detected without any compromise in accuracy or debug. This will lead to faster CDC closure.

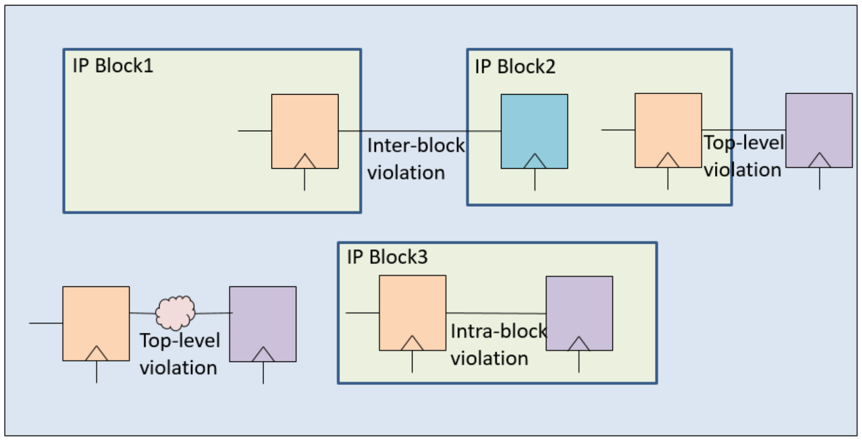


Fig 1: CDC issues example in SoC

III. EXISTING HIERARCHICAL METHODOLOGIES AND THEIR SHORTCOMINGS

Multiple hierarchical CDC verification methodologies exist in the industry, but there are shortcomings that have been detrimental to their universal adoption. In this section we will discuss the existing methodologies, the challenges associated with them, and why they fail to meet the IP-specific flow requirements.

- **IP blackbox methodology**: This is the simplest hierarchical CDC verification methodology, where IP blocks are blackboxed during SoC-level CDC verification. In this approach, the runtime of CDC analysis is reduced considerably. Also it ensures that CDC issues inside the block are not reported during SoC CDC verification, thus reducing the verification effort. However, this methodology is flawed and significantly compromises accuracy. Using this methodology, users will be able to detect the top-level CDC issues but will miss the CDC issues across the IP block interfaces. Also there might be clock or reset signals passing through the IPs. And since the IPs are blackboxed, the clock and reset tree will be broken, which will lead to incorrect results. So even though the IP blackboxing methodology allows a quick turnaround, it cannot be used for verification signoff.
- **IP graybox methodology**: In this methodology, an IP block is considered as a gray box during SoC-level CDC verification. In this approach, CDC paths completely inside the IP blocks are ignored and only CDC paths at the IP interfaces are analyzed and reported. This methodology is better than the blackbox approach in terms of accuracy as it can detect the basic CDC issues across the IP blocks. However there are some inherent drawbacks with this approach. This methodology assumes the IP blocks to be CDC clean and does not perform any validation checks on the IP interface. For example, it might be possible that during IP-level verification two clocks were treated as synchronous, whereas during IP integration asynchronous clocks were connected to them. In such a case, either the clock connections need to be corrected, or the IP needs to be re-verified with the asynchronous clocks. But a graybox methodology will miss such issues.
- **Abstract model or constraints based methodology**: The constraints based or abstract model based CDC verification methodology have been present in the industry for some time now. The underlying principle of this methodology is that during IP-level CDC verification, an abstract model in the form of constraints is generated. This constraints file is then passed on to the SoC team who use the file for SoC-level CDC verification. This approach is significantly better than the IP blackbox and graybox methodologies since it can handle the primary drawbacks mentioned above. In this approach, clock paths and reset paths can be reconstructed in an SoC-level run by using the contraints generated in a block run. Also this methodology can detect simple CDC issues across IP block interfaces and also flag and validate the block-level constraints, such as clock relationships and constant specifications. However there are multiple challenges associated with this methodology as described below.

- **Limited accuracy and debug capabilities**: These hierarchical methodologies were conceptualized with the primary objective to address capacity challenges of larger SoCs. Most of the hierarchical approaches banked on the fact that this is a faster way to run CDC verification on designs that would take long time to verify in flat mode. Hence, it was considered acceptable to produce 80% of the real issues in 50% of the flat run time with limited debug capabilities. However this assumption is flawed. A single missed CDC re-convergence bug can lead to a silicon respin and undo months of verification effort and costs. Since the abstract model is captured in the form of constraints, there is a limit to the amount of block-level information that can be captured through constraints and complex logic cannot be stored in the abstract model. So there is always a possibility of missing complex CDC issues. Also, in most cases the IP block schematic is not available in the SoC run. This lack of debug information also proves costly and ends up lengthening verification cycles. For example, users cannot debug or fix a CDC violation efficiently unless they see the actual complete path in the schematic. Partial debug increases risk of missing real issues.
- **Missing support for parameterized IP blocks**: In general, IP blocks can have different parameters and, based on the parameter values, the IP behavior changes. The same IP block can be instantiated with different parameter configurations in the SoC. In such cases, using the same abstract model for all instances will lead to incorrect results. The existing methodologies do not address this challenge. There is a need for an automated methodology which can automatically detect the configurations, intelligently use the relevant information of the IP, and ensure no issues are missed or no false violations are flagged due to different parameter configurations.
- **Limited capabilities for IP-specific requirements**: With the increasing usage of IP blocks in diverse SoCs, there is a need for a methodology where the design engineer can embed the IP integration specification along with the IP. Whenever the IP is integrated, the specification needs to be verified. The current methodologies primarily focus on verifying the CDC paths at the IP interface, but verifying the IP integration rules are not part of the methodologies. Also, since IP and SoCs can be developed and verified by completely different teams, there can be differences in environment settings and versions of tools. The existing methodologies assumes that the settings across IP and SoC CDC verification will be similar, and it is the verification engineer's responsibility to keep them consistent. But if there is a difference in the settings, it might lead to false violations or missed CDC issues. So there needs to be a methodology that can take into account these differences and take corrective action automatically wherever needed.
- **No concept of reusable CDC IP**: Another assumption of the existing methodologies is that the CDC hierarchical models can be regenerated on demand, whenever needed. There is no notion of a CDC IP that can be archived and reused whenever the design IP is reused. It's a major deterrent when users need to recreate definitions for IP blocks that are reused across projects. IP blocks need to be associated with a CDC IP that can be created just once and then reused forever without worrying about integration conditions, project scope or backward compatibility.

Hence, even though hierarchical methodologies have been around for a while, they are a long way from being absolutely accurate and merit being the default flow for CDC verification signoff.

IV. HYBRID DATA MODEL BASED CDC VERIFICATION METHODOLOGY

In this paper, we propose a systematic, accurate, hierarchical verification methodology that addresses the challenges described in the previous section and leads to faster CDC closure. The configurability and flexibility of this methodology ensures it can cater to different needs to IP to SoC CDC verification.

The basic flow of the proposed hierarchical methodology is illustrated in Figure 2a and 2b.

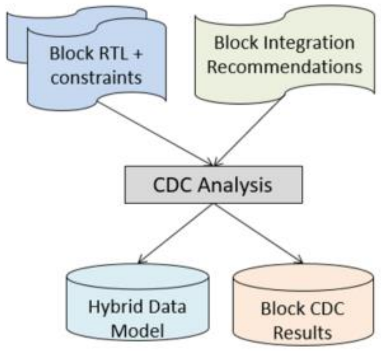


Fig 2a: Hybrid Data Model generation during IP verification

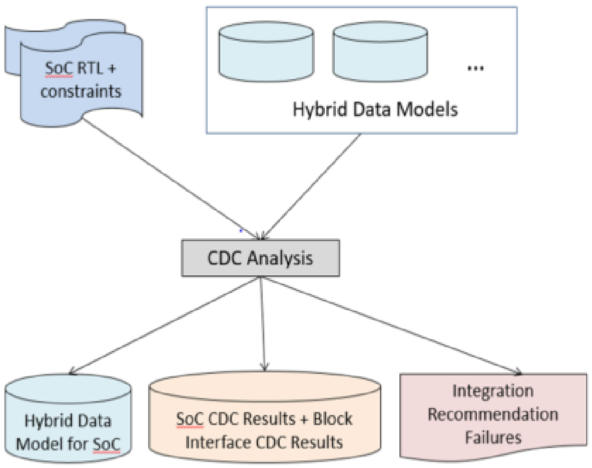
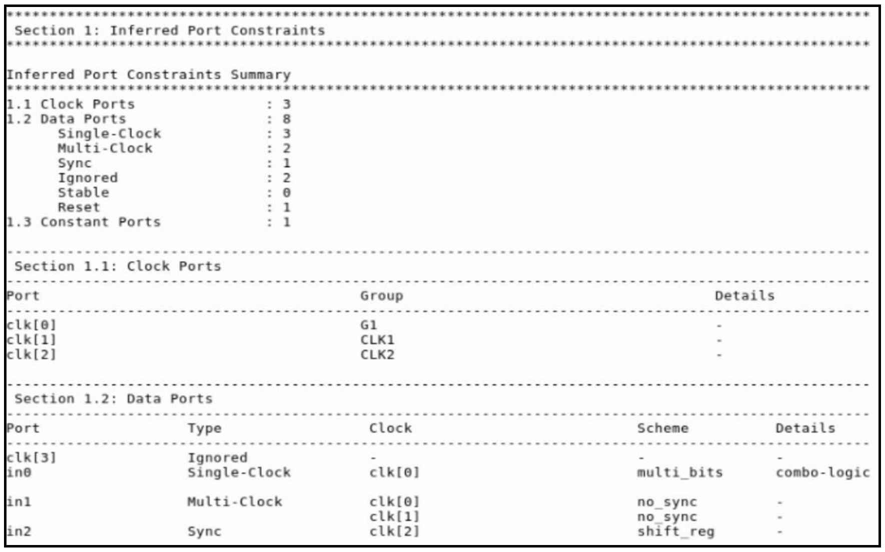

SoC RTL + constraints
Hybrid Data Models
...
CDC Analysis
Hybrid Data Model for SoC
SoC CDC Results + Block Interface CDC Results
Integration Recommendation Failures

Fig 2b: Hybrid Data Model usage in SoC verification

- Hybrid data model (HDM) generation during IP verification: During IP-level CDC verification, along with CDC results, a data model is generated. This HDM contains all necessary information of the block needed to verify and debug issues during block integration in the SoC. The IP designer can choose to generate the data model for each configuration of the IP. Also the designer can embed the integration rules in the HDM. For example, if a clock port is expected to be connected to a specific clock generator module, then such connection information can be provided during HDM generation. The IP designer can also control the visibility level inside the IP. If the designer intends to provide only the CDC model and hide the internal connectivity, then the HDM can be generated accordingly. Once the HDMs are generated for the intended configurations, they can be archived or passed on to the SoC team.
- HDM usage in SoC verification: During SoC-level CDC verification, the SoC team just needs to include the HDM files provided by the IP team. All necessary information of the IP is extracted from the HDM and used in SoC CDC analysis, and all the CDC issues across the IP block are reported to the user. Also, the user is notified about any differences between the IP-level and SoC-level settings or constraints. The integration rules specified by the IP owners are also extracted from the HDM and verified in the SoC-level CDC run. For example, if the clock port is not driven by the specified clock generator module then the violation will be flagged.
-

The HDM is the backbone of this methodology. HDM is a lightweight portable data model that can capture various information — like CDC intent of the block, schematic information of the block, integration rules, block configurations and run environment. The data model is hybrid in the true sense of the word — it stores diverse information targeted to address every possible CDC verification challenge. Even though HDM is a binary database, the user can decompile it anytime to see the internals and also modify it through constraints. Figure 3 shows an example of a decompiled HDM file.



```
***************************************************************************************************
 Section 1: Inferred Port Constraints
***************************************************************************************************

Inferred Port Constraints Summary
***************************************************************************************************
1.1 Clock Ports               : 3
1.2 Data Ports                : 8
      Single-Clock            : 3
      Multi-Clock             : 2
      Sync                    : 1
      Ignored                 : 2
      Stable                  : 0
      Reset                   : 1
1.3 Constant Ports            : 1

---------------------------------------------------------------------------------------------------
 Section 1.1: Clock Ports
---------------------------------------------------------------------------------------------------
Port                                     Group                                    Details
---------------------------------------------------------------------------------------------------
clk[0]                                   G1                                       -
clk[1]                                   CLK1                                     -
clk[2]                                   CLK2                                     -

---------------------------------------------------------------------------------------------------
 Section 1.2: Data Ports
---------------------------------------------------------------------------------------------------
Port                 Type                 Clock                         Scheme          Details
---------------------------------------------------------------------------------------------------
clk[3]               Ignored              -                             -               -
in0                  Single-Clock         clk[0]                        multi_bits      combo-logic

in1                  Multi-Clock          clk[0]                        no_sync         -
                                          clk[1]                        no_sync         -
in2                  Sync                 clk[2]                        shift_reg       -
```

Fig 3: User-readable version of HDM

V. ADVANTAGES OF HDM-BASED VERIFICATION METHODOLOGY

In this section, we will describe the advantages of the HDM-based methodology and how it addresses the challenges associated with existing methodologies.

- **Parameterized IP support**: In most cases the IP blocks have multiple parameterized configurations, each configuration having different functionality. For accurate CDC verification, it is important to generate a hierarchical model with the correct configuration which will be used in the SoC run. An SoC can also contain multiple instances of the IP with different configurations. It is the responsibility of the verification methodology to ensure that a hierarchical model with the correct configuration is plugged in for the correct instance. The proposed HDM-based methodology addresses this challenge by automatically selecting the correct configuration data model and alerts the user in case no match is found. Figure 4 illustrates this use model.

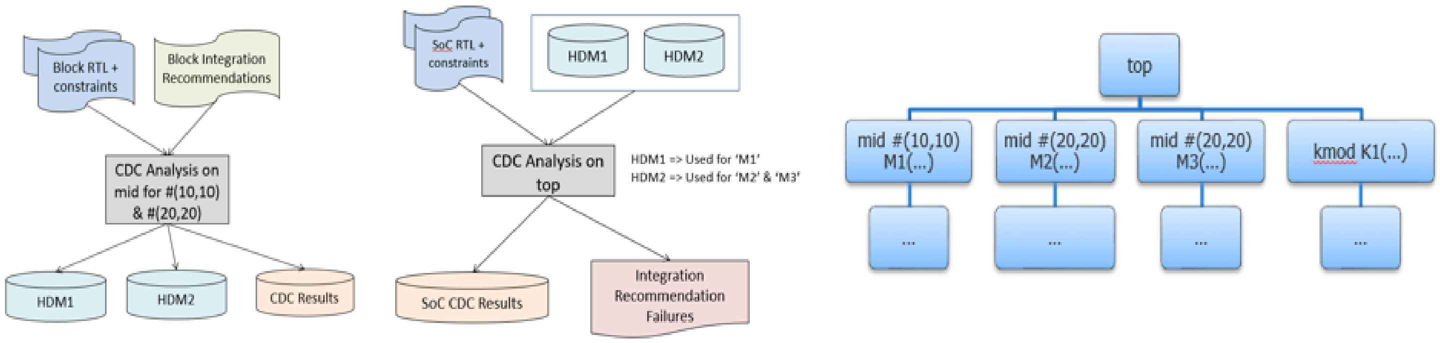


Figure 4: Use-model for design with parameterized IPs

As shown in Figure 4, the user can perform CDC analysis for each configuration of the IP and generate the HDM files. When the HDM files are included in the SoC run, the proposed methodology automatically identifies the IP configurations used in the SoC and uses the HDM files accordingly. If the HDM file is missing for any configuration, the error will be flagged to alert the user.

- **Accuracy:** The primary advantage of the proposed methodology is the data model that stores all relevant information to ensure accurate CDC verification at the SoC level without any compromise on debug. Any complex crossing that can be detected by flat CDC analysis will also be detected by this proposed methodology. Also the reporting and debug capabilities match flat run behavior. For example, if the IP block has multiple fanouts and fanins inside the block, then CDCs for each fanout or fanin will be reported accurately in this methodology. Figure 5 shows one such example where the IP input port is connected to synchronizers in the clk1 and clk2 domains inside Block IP1 and also drives output port. Using the proposed methodology all crossings are correctly reported — two synchronized crossings and one unsynchronized crossing through the output port.

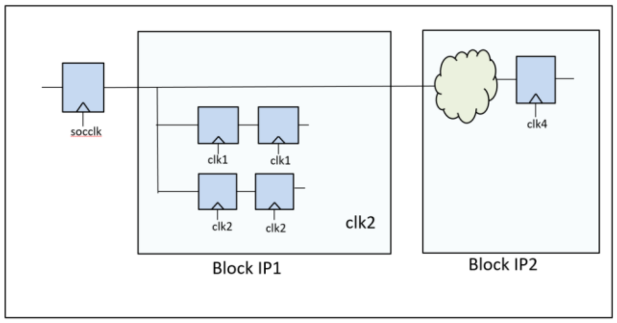


Fig 5: Accurate crossing detection by proposed HDM-based flow

Through this methodology, complex divergence and reconvergence issues can also be detected. Figure 6 illustrates one such example where a reconvergence issue spans across two different IP blocks; such issues can be identified using HDMs of Block IP1 and Block IP2.

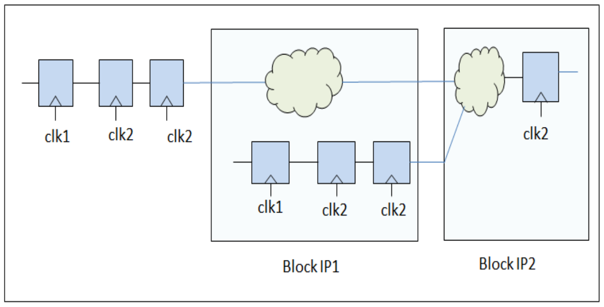


Fig 6: Reconvergence detected by proposed HDM-based flow

- **Debug capability**: In most existing hierarchical methodologies, debugging violations across the block is a challenge. In most cases the IP block is shown as blackbox in the schematic during SoC verification. The proposed methodology preserves the block schematic in the data model and displays it to the user. This ensures the user can view the complete CDC path even if part of it is inside the IP block. Figure 7 shows an example of a top-level schematic with a redundant synchronization violation across two IP blocks reported in the SoC-level run.

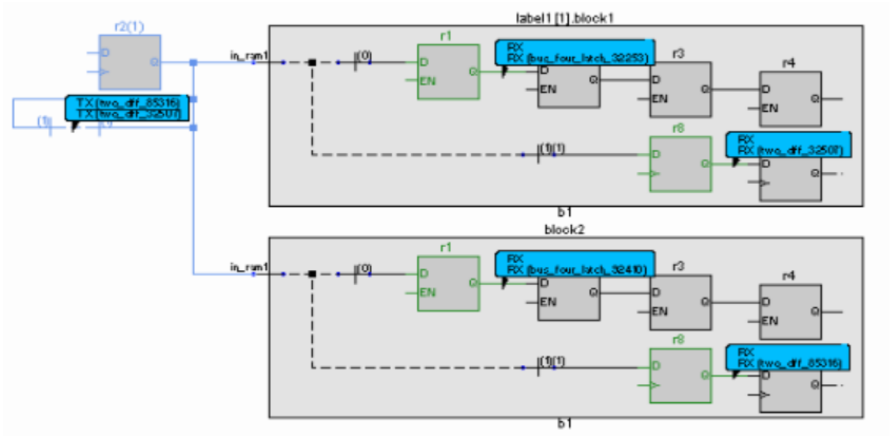


Fig 7: Redundant synchronization across IP blocks schematic using HDMs

- **Support for IP integration checks**: The proposed methodology supports integration rule checking which is necessary for IP-based flows. During CDC verification of the IP, designers can also provide specifications on properties that should hold during IP integration. The recommendations will be verified through the data model during CDC verification at the SoC level. For example, if an input port of an IP is connected to a synchronizer, then the user can specify that this port should not be driven by combinational logic after integration. Similarly if some port of the IP is expected to be connected to some specific module or specific clock domain, the IP designer can embed these rules in the HDM. The proposed methodology will verify this property during SoC CDC verification and notify the user if any of the specification rules fail. So this methodology not only ensures CDC clean IPs, but also ensures clean integration of the IPs.

  Figure 8 shows an example where a DMUX synchronizer is present at the boundary of the IP block. The DMUX synchronizer is valid only when TXDATA and TXSEL are driven by the same clock domain. The IP designer can embed this information through the "hier assume port" constraint as specified below. The proposed methodology will ensure this rule is verified when the IP HDM is integrated in any SoC.

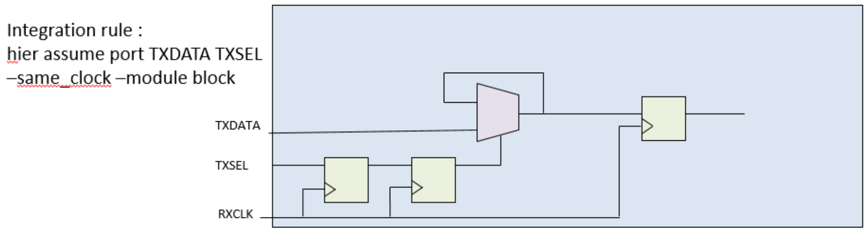


Fig 8: Example of integration rules for IP blocks

- **Configurable visibility inside IP**: The proposed methodology facilitates generation and use of each HDM with different visibility levels. For example, during generation the IP owner can control whether to expos e IP internals through the HDM or not. This ensures this methodology can be used for encrypted IPs as well as where the IP owner can pass on the HDM with the integration rules without any visibility inside the IP. In such cases the block schematic will not be available during SoC-level verification and CDC issues will be reported up to the IP block ports. Also if the IP owner has provided visibility permission, the SoC owner can control whether to use the IP internal information or not.
- **Constraints-based methodology**: The proposed methodology also provides flexibility to use a constraints-based hierarchical model. Sometimes the SoC-level verification starts before the block development is complete. In such cases generating an abstract model for the IP is not possible. In such cases, IP designers can describe the HDM through constraints which can be used during SoC verification. The constraints can also be generated from a n HDM file. So in the proposed methodology, there is flexibility to choose the use model for both IP and SoC verification engineers.
- **Reusable CDC IP**: The HDM is essentially a CDC IP that can be archived and used whenever the block is used in any design. Generally, IP and SoC development happens independently by different teams. So both teams c an use different releases or different methodologies for verification. Also after IP development, it can be used in multiple SoCs across different releases. Every time the IP is integrated, the hierarchical model does not need to be regenerated. Once the IP is finalized the HDM can be generated and archived and then can be reused across designs and across releases.

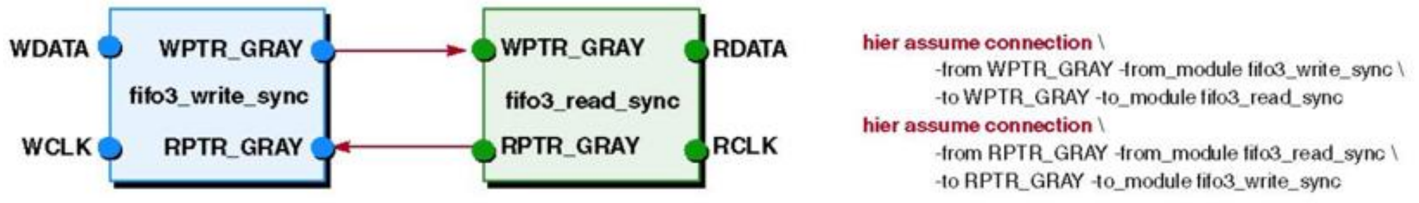
VI. CASE-STUDIES

The proposed HDM-based methodology was benchmarked on multiple SoCs with IP blocks. The results were evaluated versus traditional flat CDC verification methodology for different metrics: comparison of accuracy, performance in terms of runtime and capacity, and debug capabilities. Following are the results on one SoC with five blocks.

	Traditional Flat CDC Verification Methodology	Proposed HDM-based Verification Methodology	Gain
Runtime	3.5 hr	2.7 hr	22%
Peak Memory	20 GB	12 GB	40%
CDC Violations	75000	59000	21%

As is evident from the results, significant improvements in runtime and capacity were observed with the new HDM-based methodology. Also, the CDC violation count was reduced in by HDM-based methodology. This was primarily due to the same CDC violations being replicated in multiple instances of the same block in the flat CDC run. This redundancy was removed in the HDM-based run.

The new methodology could also catch a number of integration failures which would have been missed with prior hierarchical methods. For example, in one SoC there were instances of two IPs: fifo3_write_sync and fifo3_read_sync. For the FIFO synchronizer to work correctly the WPTR_GRAY ports of both IPs were expected to be connected to each other and the RPTR_GRAY ports were expected to be connected to each other. The integration rules were specified by the IP designers through “hier assume connection” constraints, as shown in Figure 8. There were multiple instances of these IPs in the SoC where one pair of instances of WPTR_GRAY ports were not connected. Using the proposed methodology this missing connection was detected.



```
hier assume connection \
        -from WPTR_GRAY -from_module fifo3_write_sync \
        -to WPTR_GRAY -to_module fifo3_read_sync
hier assume connection \
        -from RPTR_GRAY -from_module fifo3_read_sync \
        -to RPTR_GRAY -to_module fifo3_write_sync
```

Fig 8: Connection rules specification

VII. CONCLUSION

In this paper we presented a Hybrid Data Model that is equivalent to a CDC IP. The HDM can be generated for any IP block with multiple configurations and can be reused whenever the IP is integrated in any SoC. In this paper we also described an HDM-based CDC verification methodology that seamlessly supports IP-based flows. It eliminates the risk of missing CDC issues at the IP and SoC levels and also ensures clean IP integration by verifying the IP designer recommendations. The primary advantage of this methodology is that it meets the various requirements of an IP-to-SoC verification flow and provides configurability to modify the flow as necessary. Because the HDM methodology has debug capabilities similar to a flat CDC run, it ensures much faster analysis of CDC issues at the SoC level. Also, as observed from the case studies, the proposed approach is faster than regular flat CDC verification methodologies, addresses capacity challenges of large SoCs, reduces redundancy in CDC violations, and leads to faster CDC verification closure.

REFERENCES

[1] Clifford E. Cummings, "Clock Domain Crossing (CDC) Design & Verification Techniques Using System Verilog", SNUG-2008
[2] Ping Yeung, "*Five Steps to Quality CDC Verification"*, Mentor Graphics, Advanced Verification White Paper
[3] Chris Ka-kei Kwok, "Bridging block-level to top-level CDC verification: Hierarchical CDC verification", Designcon-2008