2020 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

Achieving Faster Reset Verification Closure with Intelligent Reset Domain Crossings Detection

Milanpreet Kaur, milan_kaur@mentor.com

Sulabh Kumar Khare, sulabh-kumar_khare@mentor.com

Mentor– A Siemens business

Abstract - The increased functionality, multiple interfaces, performance optimizations, and multi-mode operations in modern system-on-chip (SoC) designs have led to highly complex architectures of multiple primary reset sources, splitting the chip into several reset domains, each receiving different combinations of primary reset sources. In order to ensure that signals crossing complex reset domains function reliably, advanced reset domain crossing (RDC) verification, as part of comprehensive static analysis of the RTL, is imperative. An RDC verification solution must not only identify asynchronous, reset-assertion induced, critical metastability issues at reset domain crossings and glitches due to combinational resets, but also ensure reliable and accurate RDC reporting for quick verification turn around.

In this paper, we present an advanced methodology as part of a static verification tool that significantly reduces designer effort in completing RDC verification by eliminating noise from RDC results. This is achieved through proactive functional analysis of reset assertion sequences of complex combinational reset logic that drives RDC crossings. We also present a case study using real-life designs to demonstrate improvements that validate the new methodology.

Keywords—RDC; metastability; resets; SoC

I. INTRODUCTION

In recent years, higher IP reuse, increased functionality, and high-speed interfaces embedded in modern SoCs have given rise to complex reset architectures due to segregated regions that require frequent reset sequences independently. In order to bring a design to a known state, a power-on-reset strategy, also called a *cold reset,* for the entire system is deployed in the event that a power supply is unavailable. Apart from this, other reset strategies, including hardware resets, debug resets, software resets, and watchdog timer resets, are required to keep certain functionalities such as timekeeping or calendar feature up and running. System RAM and other secure blocks are required to be in active state to retain values when software requests the system into global reset, also called *warm reset*. Hence, different parts of the system have different reset requirements. The complexities of SoC designs with a large number of reset sources creating a complex reset architecture are increased with its convergence with multiple clock domains and power domains. Such complex reset architectures in such designs driving an even higher flip flop count can cause metastability issues at crossings of different reset domains.

Such reset related bugs are known to cause unpredictable reset operations or, in the worst case, overheating of the device during reset assertion. These issues are not covered by standard, static verification methods such as static timing analysis or clock domain crossing analysis. The gate-level simulations used to verify reset behavior runs too late in the design cycle. Any late-stage design changes resulting from gate-level simulations may prove expensive and, in the worst cases, result in silicon respins. Hence, a reset domain crossing verification methodology is used to catch these bugs earlier, during the RTL verification stage.

A reset domain crossing methodology as part of an automated verification flow employs several strategies to catch critical RDC bugs and help users mitigate or avoid them. However, due to the complex nature of reset architectures, multiple reset domains, driven either by primary sources or by the combinational logic of primary resets, frustrate any attempts of associating one reset per sequential element. Consequently, in large designs, the majority of sequential elements have their reset pins driven by the output of the combinational logic of several asynchronous resets and, hence, are subjected to concurrent assertion or de-assertion of several primary

asynchronous resets. This dependency may result in overlapping reset domains at the reset domain crossings and greatly complicates RDC analysis. The purpose of this paper is to stress the importance of elaborate analysis of reset assertion sequence dependency impacting RDC structures through different examples of complex RDC structures to facilitate reliable RDC bug reporting. This paper further describes in detail strategies employed as part of RDC tools to reduce noise generated by complex RDC structures, without losing the integrity and accuracy of the reset verification methodology.

II. COMPLEXITY OF RESET DOMAIN VERIFICATION ANALYSIS

A. *Reset Domain Crossings*

Let us understand the reset architecture and how the reset operation in the modern complex designs can wreak havoc to the entire chip if not subjected to verification analysis to identify the functional and structural bugs due to reset application. The reset usage for a design is primarily to bring the state of the design into a known state for simulation. Since, the reset application may introduce all types of design issues in high speed applications, it is important to define the reset strategy for the ASIC design to have optimum level of chip functioning without affecting the signal transfer. The reset strategy of modern designs mainly chooses out of two reset types:

I. Asynchronous Resets

These resets are the common and preferred ones for reset operations. These are high priority signals that bring the design to a user defined state as soon as the reset is applied without requiring the presence of clock to reset the circuit. Although, the general claim says using asynchronous resets is a sure shot way to introduce meta-stability bugs and glitches in the design considering their spontaneous nature, however, their very ability to bring the logic to a known state without having to wait for the active clock edge makes them a preferred choice for low power high speed designs. They also do not require a logic synthesis to generate the reset signal.

II. Synchronous Resets

These resets will affect the state of the flip flop on the active edge of the clock of the flip flop as they are applied as an input to the state machine. These resets are able to meet the reset recovery time and avoid glitches as they are fully synchronous to the clock of the flip flop. Recommended for designs where internal conditions generate soft resets so that clock synchronicity filters out any clock edge glitches of the reset. However, reset assertion requires clock edge for operation and may result in failure for slow clock designs. Power efficient designs with gated clocks and faster designs must also avoid synchronous resets as additional logic synthesis will increase the data path timing slowing down the design.

Hence, extensive use of asynchronous resets in ASIC or FPGA designs carries the additional risk of functional bugs at reset de-assertion i.e. when reset is released as the release might happen close to the clock edge and may violate reset recovery time, which is amount of time required between reset de-assertion and next active clock edge at the flip flop.

The above concern is irrelevant in case of reset assertion at a flip-flop as assertion renders the flop output functionally inactive, thus putting the output to a known state for multiple clock cycles. However, the higher complexities of Modern SoCs require multiple asynchronous resets targeting specific IPs or power domains. Demand for higher functionality, varied area, timing and power considerations throughout the design has led to the rise in number of localized reset domains driven by multiple asynchronous resets giving way to the rise in RDC bugs arising at the crossings where data signal travels across different reset domains.

The asynchronous reset assertion at transmitting flop causes asynchronous data change at the receiving flop without any consideration for setup and hold time requirements at receiving clock domain which may induce meta-stability at receiving logic. Such meta-stability propagates into the downstream logic throughout the design. Such issues across the crossings are termed as Reset Domain Crossings.

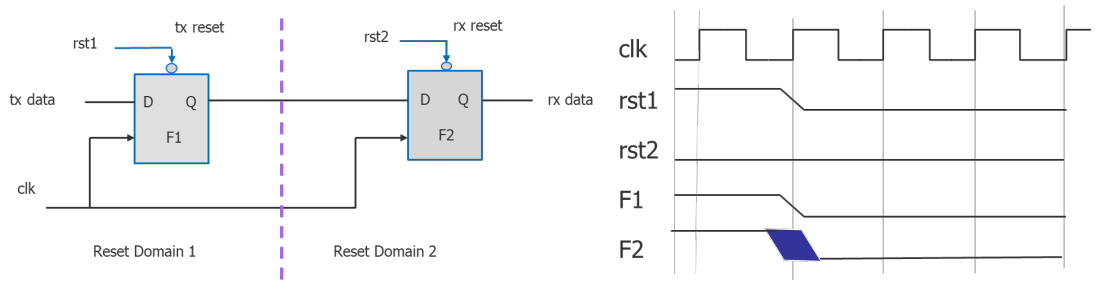

Figure 1. A reset domain crossing

Considering the random nature of such meta-stability reset assertion induced bugs even after ensuring synchronous reset de-assertion, they are extremely difficult to catch through either code reviews or simulation strategies involving formal assertions which do not provide enough test coverage at RTL level. Static timing analysis tools are ineffective as well as they address timing requirements for predetermined clocks and cannot work with asynchronous changes. Hence, an automated static methodology called reset domain crossing verification is required which must be accurate, reliable and have minimum noise levels to ensure critical reset bugs are thoroughly reviewed and taken care of by the designer.

Static analysis performs structural and functional checking on reset architecture in order to identify reset bugs like Asynchronous reset domain crossings, reset synchronizers and their re-convergence, Glitch detection etc. The functional issues critical to the accurate operations of the design are reported as violations to the user to debug. Static verification flow has been discussed by various literature sources, consider (3) explaining reset architecture structural analysis and functional analysis.

B. Multiple Resets:

As the design complexity increases, so does the reset source count and the flip flops driven by the reset sources. In order to facilitate different functional modes, primary reset sources are further modelled to generate localized reset domains. The growing complexity of reset domains generates millions of violations which mandates a sophisticated and proactive methodology to help the user manage the violations and sort through the real issues. Static reset verification is an effective and cleaner alternative to code reviews, simulation and formal verification flows and Static timing analysis, all of which are either not scalable enough to provide full test coverage to catch and diagnose sporadic RDC bugs or ineffective with asynchronous changes of data signals.

The downside of static tool is its ability to address potential issues in the design based on reset architecture analysis, which if not advanced enough, may lead to reporting a huge number of potential problems resulting in the extended manual bug review giving low return on investment. RDC Tools consistently require fine balance of catch and diagnose of potential but critical issues along with achieving low noise results by enhancing the verification flow according to the complexity of reset architecture.

The major contributor of RDC tool's noise increase is the rise of primary reset sources and as a result, combinational reset sources created throughout the design to facilitate functional modes. As explained in literature [1], CDC allows clear and clean definition of associating single clock frequency/phase with the clock domain of a sequential element and requires multiple runs for different functional modes. However, such association in RDC is not possible, as a single reset domain of a sequential element is dependent on concurrent assertion/de-assertion operations of several reset sources, either primary or combinational resets, thus complicating the RDC analysis. There are several multiple reset scenarios that demonstrate the overlapping reset domains of transmitting and receiving reset domains

Consider below scenario in Fig. 2, which has local resets as well as global resets in combination causing meta-stability issues at the output register.

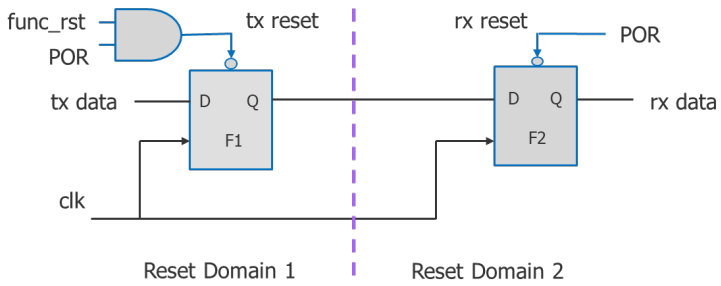

Figure 2. Reset domain with multiple, dependent async resets

F1 has local reset func_rst in addition to global reset POR which asserts during some localized fault in design. In the absence of specialized constraints to group resets that are part of the same reset domain or major design interventions, the reset domains of F1 and F2 are different resulting in a crossing. F1 reset pin is dependent on both func_rst and POR assertion sequences and hence, may result in metastability at F2.

Let us work with different reset sequences and figure out the presence of RDC issue here. Based on the schematic in Figure 2, listed in Table 1 are four reset assertion scenarios involving 2 primary async resets: func_rst and POR. In first and second scenarios, POR is both the transmitting and the receiving reset domain. In this case, there is no possibility of an RDC issue, as once POR is asserted (low) F2 will be reset regardless of the state of the other reset. However, in the fourth scenario where func_rst is the source reset domain, in asserted state and POR is the destination reset domain, not in asserted state, RDC tools will identify that F2 may go into metastability and there is a potential for RDC issue.

Table I

RDC Scenarios, multiple dependent resets assertions

S. no.	func_rst	POR	Result
1	1 → 0	1 → 0	No RDC issue
2	1	1 → 0	No RDC issue
3	1	1	No Reset Assertion
4	1 → 0	1	RDC issue

Above scenario shows the complexity in RDC analysis due to multiple reset dependencies which escapes simulation and formal tools coverage and are caught by static tools. RDC static verification tools are adept at detecting such functional issues and are able to detect millions of such violations in a large SoC design.

Traditional RDC static analysis tools follow the approach of creating a new reset domain for the combinational output of multiple reset sequences. This approach of covering multiple reset sequences leads RDC tool to generate large number of crossings in order to not miss any critical issues. However, such approach may also lead to a large number of false issues that makes filtering through the actual issues a time consuming task for the designer delaying the closure of RDC analysis and renders the tool inefficient. It may also lead to missing the critical issues which may prove to be costly causing multimillion dollar silicon respins.

Alternative design methodologies and specialized constraints are some of the advanced techniques RDC tools employ to quickly and effectively reduce the noise of RDC violations.

Based on the characteristics and functionality of the resets, a large number of resets may share the same reset domain which can be provided to the RDC tool by the user to minimize false violations reporting by the tool. This is discussed in the next section in detail. However, there is a certain class of false violations, generally overlooked by the traditional RDC tools, which include crossings where reset assertion at transmitting sequential also resets receiving sequential, making the crossing safe from meta-stability.

Consider the below scenario in figure 3, which again has local resets as well as global resets in combination at transmitting and receiving registers.

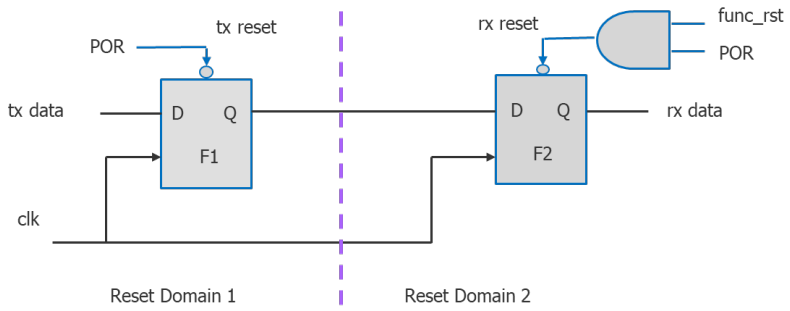

Figure 3. RDC crossing with multiple dependent async resets

F1 is receiving global reset POR and F2 is receiving local reset func_rst in addition to global reset POR. Reset func_rst asserts F2 during some localized fault in design.

Let us work with different reset sequences and figure out the presence of RDC issue here. Based on the schematic in Figure 3, listed in Table 2 are four reset assertion scenarios involving 2 primary async resets: func_rst and POR. In the first two scenarios, POR is in both transmitting and receiving reset domains. In this case, there is no possibility of an RDC issue, as in case of power failure, Power ON Reset (POR) is asserted (low) changing the state of F2 to reset state regardless of the state of the other reset. Third and fourth scenarios trigger no reset domain crossing due to F1 not going into reset state as any localized fault triggering the local reset func_rst and changing the state of receiving flop does not cause metastability at the crossing of F1 and F2.

Table II

RDC Scenarios, multiple dependent reset assertions

S. no.	POR	func_rst	Result
1	1 → 0	1→ 0	No RDC issue
2	1 → 0	1	No RDC issue
3	1	1 → 0	No Reset Assertion
4	1	1	No Reset Assertion

Above scenario shows the complexity in RDC analysis due to multiple reset dependencies and how current static RDC tools report such a crossing as violation due to combinational elements creating separate domains at transmitting and receiving sequential elements. Such false violations must be taken care of by the tool proactively

by handling complex dependencies of reset sequences to detect and exclude them from results when external specifications of reset grouping cannot resolve the newly created reset domains.

III. PROPOSED METHOLDOLOGY

An RDC Verification methodology is designed to identify any critical RDC paths through structural and functional analysis and provide guided methodology to mitigate such paths with minimal effort. Containing the bugs early on is useful in stopping them from propagating to silicon and avoid huge losses due to high-cost respins.

However, the efficiency of the tool is hampered if the verification flow is not time efficient and delays the issue debugging process for users by dumping inaccurate results. There are various ways to leverage the capabilities of RDC tool by utilizing some below mentioned strategies to optimize reset architecture.

- Reset domain grouping by designer
- Reset Sequencing specified by designer

A. Reset domain grouping by designer

Template based RDC analysis generates noisy and inefficient results and modern RDC tools aim to involve user in simplifying the complex reset architect by providing specialized constraints for user to group resets as part of the same domain based on their basic characteristics and functionality implementation. In order to group the rests under the same reset domain properties such as Polarity, Synchronicity to clock domain of flip flop (Synchronous/Asynchronous), flip flop reset value (set/reset) and the top reset name must be same. In addition to this, based to functionality; resets which, although not connected to the same source pin, assert together through some external source. Delayed reset assertion scenarios where assertion of one reset triggers the assertion other reset immediately or with some delay (acceptable delay which does not propagate incorrect data into the design) and vice versa, may be grouped under the same reset domain if the designer deems necessary.

Such grouping of resets under the same reset domain can help the tool in simplifying the reset architecture and cause significant reduction of false reset domain crossings reported earlier by the tool.

B. Reset Sequencing specified by designer

Apart from specifying similar resets under a single reset domain, reset sequencing through user specified reset orders are also utilized to simplify the reset architecture. Designer can specify the reset assertion sequence where assertion of a reset, particularly a hard reset, triggers the assertion of a specific, more localized reset. However, in this case, vice versa is not true and localized reset cannot trigger the assertion of hard reset. Such assertion orders help marking the crossings where the assertion sequence applies, as safe crossings as meta-stability does not propagate in this case.

Consider the below scenario where F1 and F2 both receive combinational resets, thus making the reset logic dependent on several reset sequences.

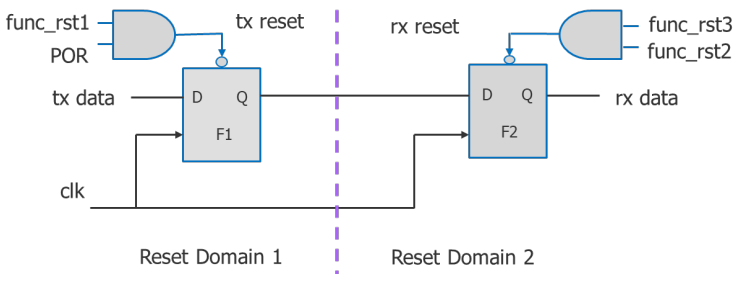

Figure 4: Reset domain crossing with combinational resets

Below are the user defined specifications:

- POR and func_rst2 share the same reset domain.
- func_rst3 and asserts before func_rst1

Below the table summarizes the impact of using reset domain specifications over crossings result:

Table III

RDC results, multiple asynchronous reset sequences

S. no.	func_rst1 (Tx reset)	POR (Tx reset)	func_rst3 (Rx reset)	func_rst2 (Rx reset)	Result
1	1 → 0	1 → 0	0	1 → 0	No RDC issue
2	1 → 0	1	0	1	No RDC issue
3	1	1 → 0	1/0	1 → 0	No RDC issue
4	1	1	1 / 0	1	No Reset Assertion

First three scenarios show Tx reset assertion (due to either func_rst1 assertion or POR assertion or both) which does not cause any RDC issue due to Rx reset assertion at the same time due to user defined assertion sequence ordering. The last scenario does not cause any Tx reset assertion, hence no RDC. Therefore, it is safe to say the user specifications are able to reduce any false violations that have been reported due to complex reset domains created internally due to soft reset implementations.

However, in case of design implementations which do not have resets sharing the same reset domains, RDC tools may still report violations which are functionally safe crossings due to complex reset implementations resulting in reset domain overlapping across crossings.

In this sub-section, we list down some of the RDC structures found in SoC designs with overlapping reset domains across crossings:

1. All resets sequences in source flop reset domain impact destination flop reset domain.

Consider the schematic in figure 4, where reset assertion at source sequential is clearly asserting destination sequential, making the RDC safe. However, static RDC analysis reports it as a violation. New methodology identifies such scenarios by structural and functional analysis of the common reset sequences impacting the reset domains of both the flops.

To elaborate further, consider the schematic in Figure 5. Metastability is induced at receiving flop F2 by asynchronous assertion of source flop F1 receiving single reset POR which also asserts F2 simultaneously, given the same polarity of the common reset func_rst1 reaches both the flops.

Above case is a false RDC issue and is pruned out during new methodology of RDC analysis.

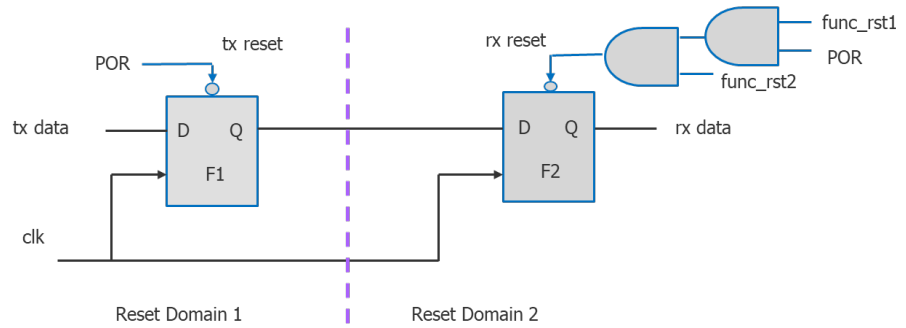

Figure 5. Dependent resets RDC crossing, all source resets impact destination

2. *Some reset sequences of Source flop reset do not impact destination flop.*

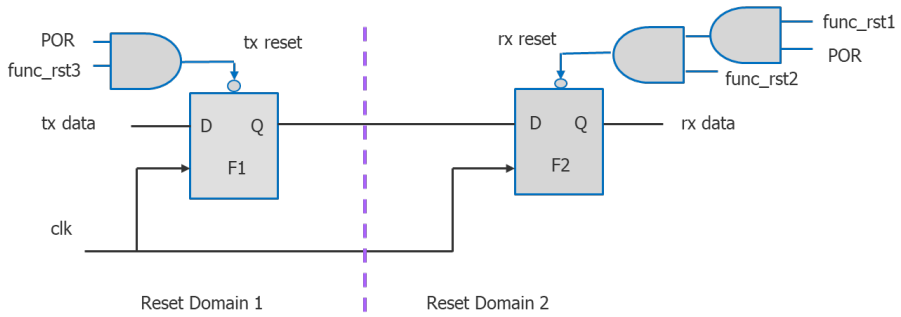

Figure 6. Dependent resets RDC crossing, not all source resets impact destination

In the above scenario of fig 6, localized reset domains implemented for a specific reset functionality has reset assertion sequences specific only to the source flop and does not impact destination sequential. Any data change in source flop F1 due to such non-common resets assertion, such as func_rst3, may induce meta-stability if all the reset sources of destination flop F2 are not activated despite safe crossing due to reset assertion of common reset POR. However, assertion of func_rst3 may lead to metastability in case common resets and Rx resets are not in assertion state. New methodology ensures such, seemingly elusive but a potentially dangerous issue, is caught as a violation and reported to the user for review.

However, the review of the crossing reveals that it is a safe crossing if the reset sources func_rst1 and func_rst3 are sequenced in a way that assertion of func_rst3 is always triggered after func_rst1. In other words, func_rst3 cannot assert before func_rst1 and any metastability induced by the non-common reset source at transmitting flop, func_rst3 can be blocked from further propagation in the design by reset sequence timings. The new methodology proactively identifies user defined reset ordering at primary reset sources along with the dependency of resets and propagates them to identify a false violation.

The new methodology of optimizing the reset architecture helps weed out false reset domain crossings and significantly reduces the debugging time for designer to review RDC results with the help of debug aids. This methodology also supports the debug aids in the form of reset order suggestions in order for the user to optimize the reset architecture setup and help the tool provide accurate and reliable results. It provides a cleaner, less noisy report to the designer to invest efforts in resolving real, critical RDC bugs. Thus, the above methodology makes the RDC tool more efficient and reduces RDC verification closure time.

IV. CASE STUDY

The proposed methodology was used on a highly complex real SoC with more than 1.8 million registers, and 5 RAMs. RDC verification tool with the new methodology was used that identified around 287 reset domains in the design, which consisted of 31 asynchronous domains defined by the user as well as 233 asynchronous reset domains inferred due to combination of resets. Out of these, around 23 synchronous reset domains were not analyzed for reset domain crossings.

A. The first run analyzed data path crossing asynchronous reset domains without any ordering or grouping information from the user and reported around 90k RDC crossings.

B. After applying reset grouping, the inferred combinational reset domains were dropped to 158. Reset order was also given explicitly for 90 pairs which increased the ordering pairs to 156 reset pairs when used associatively. Below are the RDC results changes after applying reset grouping and reset ordering information.

Table IV

Reset Domain Crossings without grouping and ordering information	**Number of crossings**
RDCs having source and destination registers in different asynchronous reset domains ***without*** user constraints methodology	57688
RDCs having source and destination registers in different asynchronous reset domains ***with*** user constraints	34562
Ordered RDC paths (based on sequencing information)	23126

Using the new methodology along with grouping and ordering information, around 34% RDC issues were filtered out solely based on dependency of reset sources and around 20% RDC violations converted to ordered reset domain crossings.

Table IV

Reset Domain Crossings with grouping and ordering information	**Number of crossings**
RDCs having source and destination registers in different asynchronous reset domains ***without*** proposed methodology	34562
RDCs having source and destination registers in different asynchronous reset domains ***with*** proposed methodology	22811
Ordered RDC paths (based on sequencing information)	27650

V. CONCLUSION

The proposed automatic technique significantly improves the quality of results for static RDC analysis, and it reduces the time required to identify RDC issues on Data paths crossing reset domains. This methodology identifies only the relevant unsafe RDC paths and helps users focus on verifying them effectively. Several scenarios depicted in the paper ensures no critical path is missed and false crossings are reported. Advanced techniques utilize reset ordering information as well to simplify the reset architecture and enhances the tool capabilities.

Validation on real SoCs confirms that the proposed flow and techniques are practical and must be applied on modern designs as part of static RDC methodology to prevent chip-killing reset crossing issues before they are sent for gate level analysis.

VI. REFERENCES

[1] Yossi Mirsky, “Comprehensive and Automated Static Tool Based Strategies for the Detection and Resolution of Reset Domain Crossings”, DVCON

[2] Chris Kwok, Priya Viswanathan, Ping Yeung, “Addressing the Challenges of Reset Verification in SoC Designs”, DVCon US, 2015

[3] Akanksha Gupta, Ashish Hari, Anwesha Choudhary, “Systematic Methodology to Solve Reset Challenges in Automotive SoCs”, DVCON Europe 2019