On analysis of RDC issues for identifying reset tree design bugs and further strategies for noise reduction

Farhad Ahmed, Lyle Benson, Manish Bhati,
Siemens Industry Software Inc., farhad.ahmed@siemens.com;
Siemens Industry Software Inc., lyle.benson@siemens.com
Siemens Industry Software Inc., manish.bhati@siemens.com;

Abstract – Reset tree checks should be viewed thoroughly before reset domain crossing analysis. Static verification tools have many checks for reset tree analysis. This paper discusses the usage of non-resettable registers (NRRs) in reset paths. NRRs can cause metastability in the reset paths and hence thorough verification is a must. The paper discusses reduction of false failure reporting noise strategies in RDC analysis. Stable paths and functional false paths are the focus of the discussion in noise reduction, and we discuss various scenarios and how static verification tool should report these paths. A large semiconductor company we partnered with on this project.

Keywords: Reset Domain, Reset Domain Crossing (RDC), Clock Domain, Clock Domain Crossing (CDC), Reset tree, Clock tree, static reset tree checks, static clock tree checks, non-resettable register (NRR), functional false path).

I. INTRODUCTION

Static verification, including Clock Domain Crossing (CDC) and Reset Domain Crossing (RDC), is imperative to finding issues in complex SoCs. A decade ago, static verification technologies were a "nice to have." Today, in an environment of ever-increasing design complexity, static verification is now so non-trivial that design and verification teams along with EDA vendors continue investigating newer checks and methodologies for static analysis. The cost and schedule impact of design issues caught late in the design cycle have made engineers focus on RDC analysis of RTL designs. RDC analysis is challenging in a sense that it involves careful consideration of design reset strategies. Fixing RDC bugs properly often requires changing the reset architecture which can be very costly at late stage in the design cycle. Additionally, a poorly architected reset strategy can lead to reset bugs escaping to silicon thereby leading to prejudicial wastage of thousands or even millions of dollars, design re-spins and project schedule slippage.

It is often observed that engineers jump past analyzing RDC results without reviewing the setup checks. Setup check results must be analyzed. A RDC analysis run is initiated by compiling and elaborating the design thereby performing a detailed analysis of clock and reset trees. A good RDC analysis engine will do the analysis pessimistically so that it covers all the domain crossing signals and related issues, if any. Pessimism in the analysis may lead to creation of new clock and reset domains which can increase the noise in the results. Engineers can control the level of pessimism by using proper constraints. RDC analysis tools carve out analysis on clock and reset trees and provide a set of pointed/focused issues a designer can focus on before doing analysis on the actual domain crossing signals. There are many checks that need to be performed on the clock and reset trees and this paper will discuss few such issues related to reset trees.

We have seen design engineers regularly use non-resettable registers (NRRs) in RTL designs. Resettable registers may be used on the periphery of the design while NRRs are generally used on internal paths. Using NRRs have many advantages, including reduced area and lower power consumption. Using non-resettable registers also comes with some challenges. We will highlight one such disadvantage of non-resettable register usage on reset paths.

RDC analysis results can be noisy because of the previously mentioned inherent pessimism in the analysis. RDC analysis methodologies need to provide a noise management strategy which requires minimal extra work for engineers. If noise management is not done properly it may lead to serious RDC issues being incorrectly masked, or not reported. We have investigated RDC analysis results on many designs and have found that often the analysis results are noisy because the analysis engine may not account for stable constraints applied on various signals on clock, and control paths. Using knowledge of system behavior and architecture, an engineer may apply constants and stables on certain signals; configuration registers are usually perfect examples of stable signals. These registers usually don't toggle during the functional operation of a chip and hence are not candidates for CDC/RDC analysis. It is expected that EDA tools consider these signals to be static and report paths associated with these signals as safe crossings. For RDC analysis we have observed that the tools report these "safe" paths as "unsafe" leading to noisy results. This paper investigates how RDC tool should report such paths as safe RDC crossing.

Engineers tend to use constraints (i.e. setting false paths) to eliminate paths from analysis completely, but this is no better than a waiver and can result in bugs. It is our recommendation that design rules should strictly disallow usage of such constraints. It is our belief that any RDC tool should automatically recognize functional false paths and report them. Functional false path reporting by the tool has many advantages. The first and foremost is that all the domain crossing paths are analyzed by the tool. EDA tools may apply formal verification technology to report these types of paths (even if unconstrained by the user). Doing so provides confidence that all the paths are analyzed. Additionally, engineers can make necessary changes to the design at an early stage of design development to address domain crossing issues identified by the tools. The proposal we support in this paper is to analyze these paths under a different RDC scheme.

II. NRRs ON ASYNCHRONOUS RESET PATHS MAY LEAD TO RDC ISSUES IN THE DESIGN

Engineers sometimes use NRRs on asynchronous reset paths. There are many reasons engineers use this design technique, adding delays on a reset path and using NRRs to reduce power and area are two examples. This design style incurs a risk of creating RDC issues between a transmit source with an asynchronous reset and the receive destination having the same asynchronous reset with NRRs on the reset path. During RDC, EDA tools generally don't flag problems between the source/destination registers because the delays caused by the NRR's on the reset path are not taken into. EDA tools tend to assume that both transmit source and receive destination registers belong to the same reset domain; the path is ignored. To correctly address this issue, engineers need to define a new reset on the last NRR output. The tools will then correctly flag a RDC violation between the transmit source register and the receive destination register. It is a cumbersome, time consuming and tedious process for engineers to define a new reset in the setup file for every path that exists in the design. We present two such cases below.

Case 1: Consider the schematic in Fig. 1. Here the receive destination register "rx" has NRRs on its asynchronous reset path. An RDC path exists between the transmit source register "tx" and receive destination register "rx". The NRR path delays on the reset path of the "rx" register, may cause the reset at the "rx" register to assert asynchronously relative to the "tx" reset.

If we do not account for the time delays from the NRR's (f2, f3) the data path between "tx" and "rx" could be incorrectly identified as a synchronous reset data path. We must take the NRR delays into account, which manifests a behavior of the "rx" reset asserting after the "tx" reset asserts. Which is a condition where metastability could be sampled across the data path.

One attempted solution could be to use a reset ordering constraint, but this type of constraint assumes that the "rx" reset asserts first before the "tx" reset. In this case reset ordering would not address this circuit condition. Because the "rx" reset would assert after the "tx" reset due to the time delays from the NRRs in the reset path.

This is an example of why the presence of the NRRs in the reset path should be addressed. They should be reviewed and understood in the context of the design's functional behavior and the time delay caused by the NRRs creating an asynchronous reset behavior must be addressed.

To aid the design engineers, an analysis check is required to identify these conditions (e.g. NRRs in the reset path) so a design team can address this condition, and remove the risk of metastability sampling due to the asynchronous reset behavior caused by the delays in the reset path.

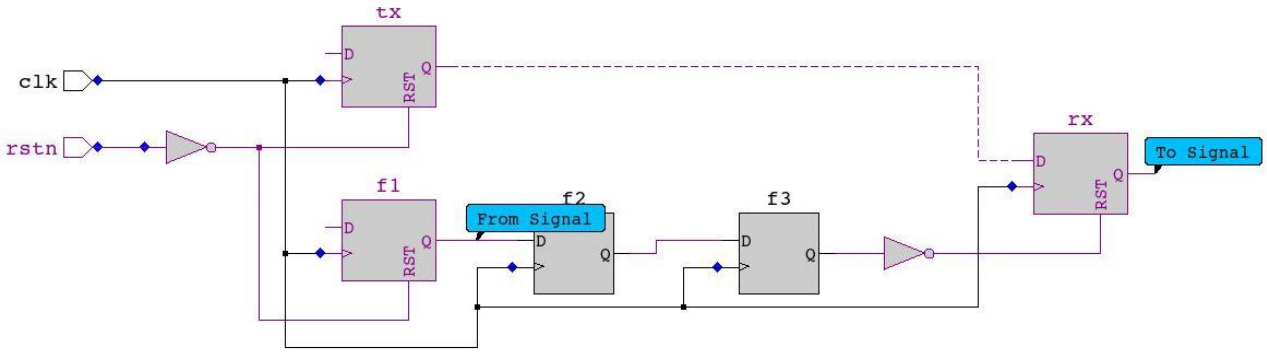

Figure 1 NRR on Reset Path with same async reset on both TX and RX sides

Case 2: Consider the schematic in Fig. 2. Here the receive destination register "rx" is driven by an asynchronous reset "rstn2" and source register "tx" is driven by an asynchronous reset "rstn1". Destination register "rx" has NRRs on its asynchronous reset path. Even with correct reset order constraint, this may cause RDC metastability problem between source and destination registers due to the "rx" reset path delay.

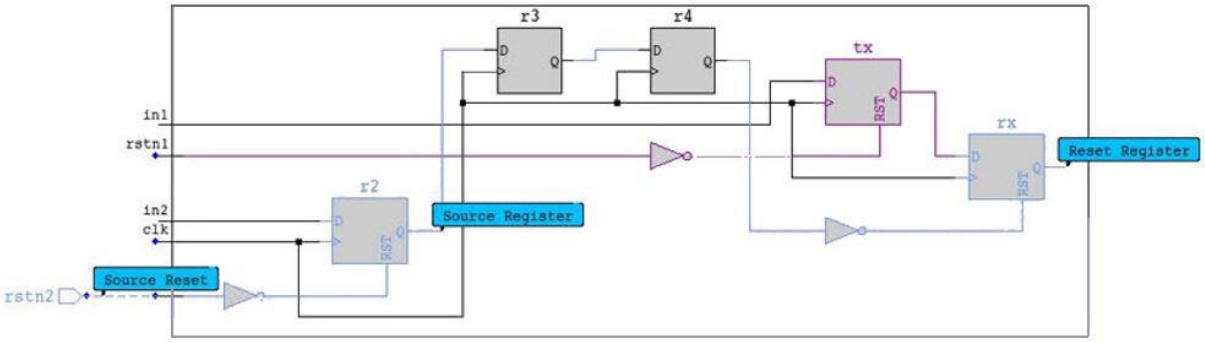

Figure 2 NRR on Reset Path with different async reset on both TX and RX sides

III. NOISE MANAGEMENT IN RDC ANALYSIS

RDC analysis on SoC designs can have thousands or even millions of violations, some of them may be false violations. Engineers often use wild cards and regular expressions to write generic waivers which may be applied to many violations at the same time. This is a clever way of writing waivers or constraints, but it may also lead to waiving real and potential issues. For this reason, constraints are always preferred over waivers. For an RDC path, engineers generally constrain the Rx data and clock paths through isolations or through specification flows where they mark or specify the receive register clock as "off".

Constant and stable constraints applied on control paths (e.g., register enables) and clock paths (e.g., clock pins or clock nets) of the source transmission register in the RDC path are often not accounted for by an RDC analysis tool and hence a user may see noise. It is important to understand that when a user disables a register via an enable pin using a constant (most often 0 value) as a constraint for RDC analysis, the user is generally referring to the fact that the register is at some known state and will not sample the value at input D pin. RDC analysis is supposed to be done on the operational modes of a design which happens after reset sequencing. This implies that the register will

be either at set value by virtue of set pin, or at reset value, by virtue of clear pin. Hence the assertion event at the reset signal (set/clear) will have no effect on the register whose enable pin is marked constant by a constraint. This means that if the register is a source transmission register in an RDC path, this will be a safe crossing and EDA tool should report it as a safe RDC path. RDC analysis tools mark these paths as safe. Here the RDC tool infers the stable path and thus reports it in a separate RDC category/scheme. This improves the quality of results and hence the results are less noisy.

Similar logic applies to stable or constant constraints applied to clock paths. When the operational mode of the design is under analysis, the user wants to set a constraint during the analysis that the register doesn't sample any value. We present three such cases below.

Case 1: Consider the schematic in Fig. 3. The "en" pin of the source transmission register is tied to a constant "0", making this register stable. RDC analysis tool infers these types of paths as stable and puts them in a separate RDC scheme.

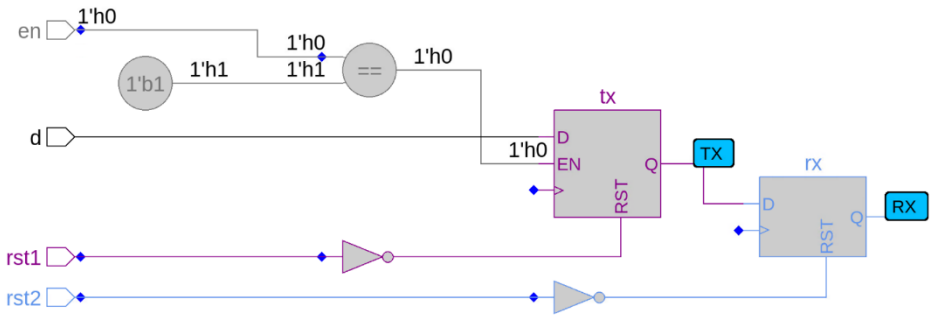

Figure 3 Constant 0 applied at control path

Case 2: Consider the schematic in Fig. 4. The clock pin of the source transmission register, "clk1" is tied to a constant "0", meaning this register will not sample any value. Hence, the "tx" register is in stable condition. RDC analysis tool infers these types of paths as stable and puts them in a separate RDC scheme.

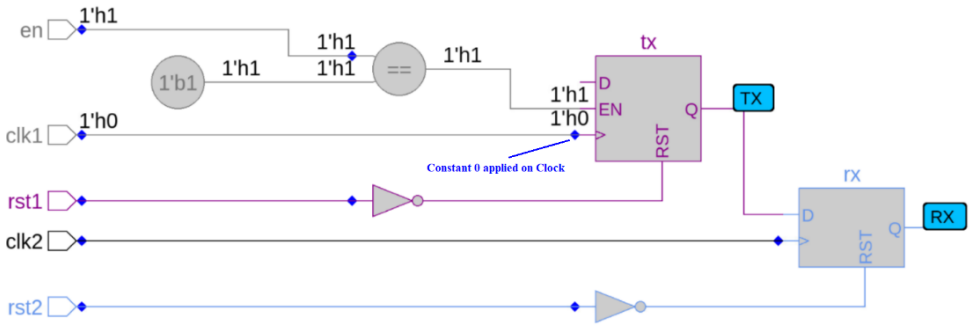

Figure 4 Constant 0 applied on clock

Case 3: - Consider the schematic in Fig. 5. Clock pin of source transmission register, "clk1" has been marked stable via constraint. The source transmission register is not going to register any value as the clock is not switching. This also means that the source transmission register is at stable value. RDC analysis tool infers this path as stable and puts it in a separate RDC scheme.

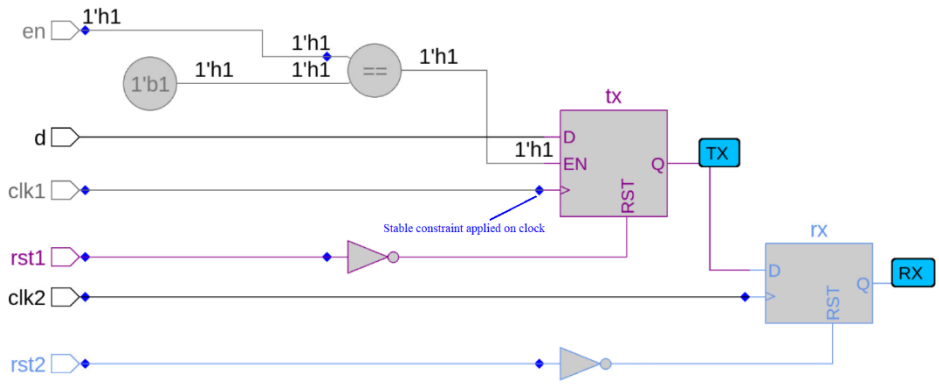

Figure 5 Stable constraint applied on clock

IV. AUTOMATIC RECOGNITION OF FUNCTIONAL FALSE PATH IN RDC

Engineers writing RTL often don't realize that some paths, which may cause RDC paths between a transmit source and a receive destination, are functionally false. These paths may not cause any functional failure in the design, and these false paths can be ignored in functional verification. Synthesis tools also don't have a way to identify these functionally false paths, but during RDC analysis, these paths will appear as RDC violations if a reset domain crossing exists between the transmit source and receive destination. These RDC violations are considered noise in RDC analysis result. Depending on how many such paths exist in a design, reviewing and addressing these issues can be an inefficient, time-consuming effort, that wastes valuable time. In this paper we will show how a RDC tool will be able to identify these paths automatically as functionally false path and identify them as safe paths. This reduces the amount of time and effort an engineer must spend analyzing and debugging RDC results.

Case 1 – Consider the schematic in Fig. 6. There is an RDC crossing between the transmit source "t1" and the receive destination "out". These two registers are driven by same clock but different asynchronous resets. Looking at the path between the two registers, there are multiple muxes on the path. When the select pin is set to "0" for the first mux the "D0" pin is selected which is connected to the output pin of the transmit source register "t1". For the second mux, the "D0" pin is not connected to the "t1" register but is connected to a different transmit source "in2". When the select pin is set to "1" for the first mux, "D1" is selected which is set to a constant "0". Hence path between "t1" and "out" can be considered as a functionally false path and no RDC violation should be reported. If a design has hundreds or even thousands of these functionally false paths, engineers won't spend their time unnecessarily debugging them. Solutions without this capability will report these paths as domain crossing errors that will require time and effort to debug.

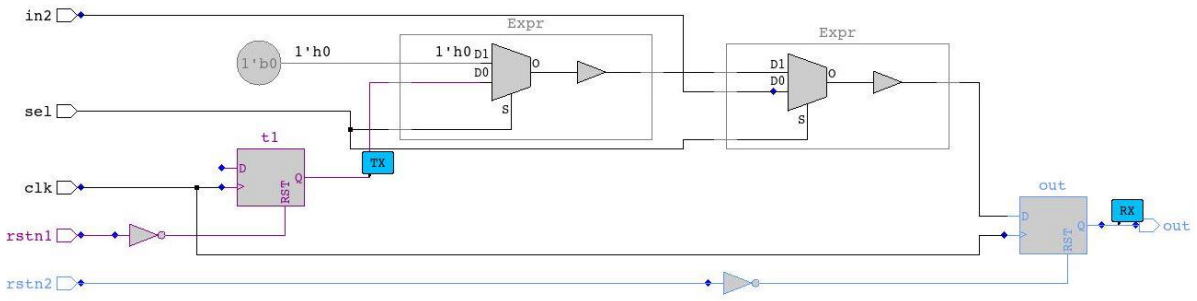

Figure 6 Functional False Path for RDC

Case 2 – Consider the schematic in Fig. 7. There are two possible RDC crossings between the transmit source "d1", the receive destination "s0" and the transmit source "d2" and receive destination "s0". All the registers are driven by the same clock. But the "d1" and "d2" registers are driven by the asynchronous reset "rstn1" and the "s0" register is driven by a different asynchronous reset "rstn2". If we write the Boolean expression for the combinatorial logic between the transmit source and receive destination registers, it is ((!din1 & din2) + (din1 & din2)). This expression can be written as ((!din1 + din1) & din2). Which can further be simplified as "din2" only. So, the RDC path between "d2" and "s0" is a valid RDC path that needs to be evaluated. The RDC analysis tool classifies the path between "d1" and "s0" as a functionally false path.

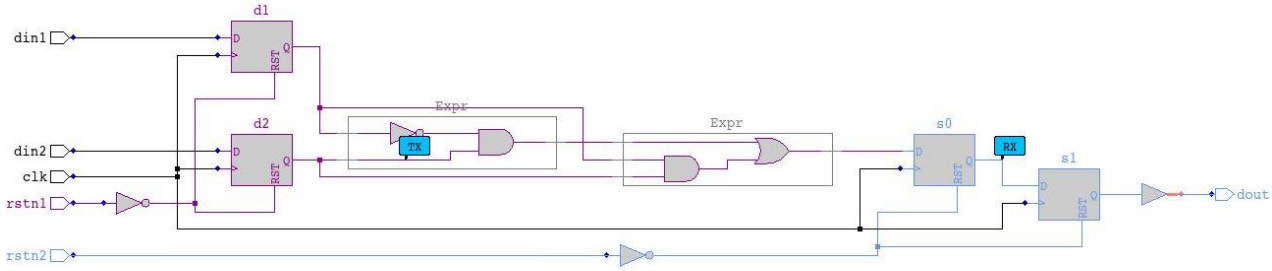

Figure 7 Functional False Path for RDC

VI. CONCLUSION

In this paper, we have highlighted specific issues that must be addressed by designers to improve RDC results and to ensure the highest fidelity in identifying real design issues. The use of NRR's in a design have many benefits, however it can come with the risk of metastability. Identifying and addressing the presence of NRRs in the reset tree are critical to identifying issues for analysis. When NRRs are present in the reset tree, they pose a risk due to timing delays in the reset tree that can manifest themselves by creating an asynchronous reset behavior even though the reset at the TX register, and the RX register are sourced from the same reset tree.

Evaluating the results from RDC analysis can take a significant amount of time to both understand, and recognize the specific fault highlighted in the identified paths. The benefits, of the analysis tool reducing the number of paths to review, through FFP detection, and stable path inference reduces the time spent by the reviewer, while allowing a more directed focus on the identified, real issues that should be addressed in the design.

VII. RFERENCES

[1] IEEE Std 1800™-2017, IEEE Standard for SystemVerilog Unified Hardware Design, Specification and Verification Language.

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

[3] Yossi Mirsky, "Comprehensive and Automated Static Tool Based Strategies for the Detection and Resolution of Reset Domain Crossings", DVCon US, 2016.