2023 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION INDIA

Techniques to identify reset metastability issues due to soft resets

Reetika – Siemens EDA DI SW
reetika.nln@siemens.com
Sulabh Kumar Khare – Siemens EDA DI SW
sulabh-kumar.khare@siemens.com

***Abstract*- Modern SoCs are equipped with complex reset architectures to meet the requirements of high-speed interfaces with increased functionality. These complex reset architectures with multiple reset domains, ensure functional recovery from hardware failures and unexpected electronic faults. But the transmission of data across sequential elements that are reset by different asynchronous and soft reset domains can cause reset domain crossing (RDC) paths, which can lead to metastability. This metastability can cause unpredictable values to be propagated to down-stream logic and prevent a design from functioning normally. A proper reset domain crossing sign-off methodology is required to avoid metastability and other functional problems in chip designs. In this paper, we present a systematic methodology, as a part of static analysis, to intelligently identify critical reset domain bugs associated with soft resets. A soft reset is a mechanism that initiates a controlled reset within the system without fully powering it off.**

Keywords— Asynchronous resets; soft resets; reset domain crossings; metastability

I. INTRODUCTION

In recent years, the complexity of designs is increasing, resulting in a huge rise in electronic components like processors, power management blocks, and DSP cores. To meet low-power and high-performance requirements, system on chip (SoC) designs have been equipped with several asynchronous and soft reset signals. These reset signals in the design, help to safeguard software and hardware functional safety as they can be asserted to speedily recover the system onboard to an initial state and clear any pending errors or events. However, with multiple asynchronous reset sources in a design, arises multiple reset domain crossings (RDC) paths, that can lead to systematic faults and hence cause data-corruption, glitches, metastability or functional failures. These issues are not covered by standard, static verification methods such as clock domain crossing (CDC) analysis. Therefore, a proper reset domain crossing verification methodology is required to prevent errors in the reset design during the RTL verification stage.

By definition, a reset domain crossing occurs when a path's transmitting flop has an asynchronous reset, and the receiving flop has a different asynchronous reset than the transmitting flop or has no reset.

Fig. 1a illustrates the simple RDC problem between two flops having different asynchronous reset domains. The asynchronous assertion of the rst1 signal immediately changes the output of Tx flop to its assertion value. Since the assertion is asynchronous to clock clk, the output of Tx flop can change near the active clock edge of Rx flop, which can violate the set-up hold timing constraints for flop Rx and, therefore, Rx flop can go into a metastable state. Metastability is referred to as a state in which the output of a register is unpredictable or is in a quasi-stable state. Fig. 1b illustrates the RDC problem from a flop with asynchronous reset domain to non-resettable register (NRR), which be definition does not have a reset pin. An RDC path with different reset domains on the transmitter and receiver does not conclude that the path is unsafe, and similarly, a RDC path having same asynchronous reset domains on the transmitter and receiver does not necessarily imply that it is a safe path as the issue may occur due to soft resets (an RDC path through the transmitter to the receiver register in which metastability can propagate is termed an unsafe path, on the other hand, when there is no metastability propagating in the path, it is deemed to be safe). The different soft resets in a design can induce metastability in designs and hence cause unpredictable reset operations or, in the worst case, overheating of the device during reset assertion.

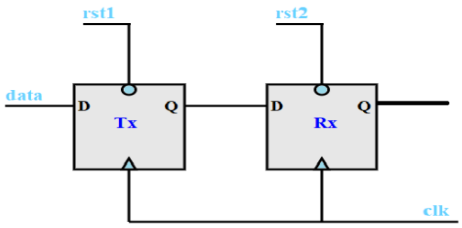

Fig 1a: RDC between Tx flop and Rx flop from asynchronous reset rst1 to asynchronous reset rst2

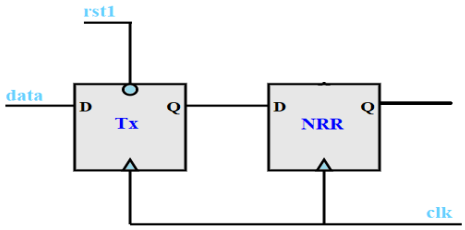

Fig 1b: RDC between Tx flop and NRR flop from asynchronous reset rst1 to non-resettable register

II. SOFT RESET ANALYSIS

A soft reset in an internally generated reset (register/latch/black-box output is used as a reset) that allows the design engineer to reset a specific portion of the design (specific module/subsystem) without affecting the entire system. Designer engineers frequently use a soft reset mechanism to reset/restart the device without fully powering it off, as this helps to conserve power by selectively resetting specific electronic components while keeping others in an operational state. A soft reset typically involves manipulating specific registers or signals to trigger the reset process.

Soft reset is a common technique used during the design phase to quickly recover from a problem or to test a specific area of the design. This can save time during simulation and verification by allowing the designer to isolate and debug specific issues without having to restart the entire simulation. Soft resets are also implemented as part of fault-tolerant systems to help recover from transient faults or errors. When a fault is detected, the soft reset is triggered to restore the system to a functional state, ensuring system reliability and availability. Fig. 2 illustrates a simple soft reset with its RTL to demonstrate that SoftReg is a soft reset for flop Reg.

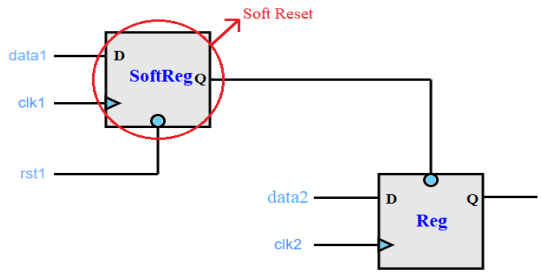

Fig 2: SoftReg is a soft reset for register Reg

```
always @(posedge clk1 or negedge rst1)
    if (!rst1)
        SoftReg <= 1'b0;
    else
        SoftReg <= data1;

always @(posedge clk2 or negedge SoftReg)
    if (!SoftReg)
        Reg <= 1'b0;
    else
         Reg <= data2;
```

The following analysis will present a systematic methodology to identify the RDC's with different soft resets that are unsafe, even though the asynchronous reset domain is the same on the transmitter and receiver ends. Also, with enough debug aids, we will identify the safe RDC's (safe from metastability only if it meets the static timing analysis) having different asynchronous reset domains that would help to avoid silicon failures and minimize false crossing results.

Techniques to identify metastability issues due to soft resets:

A. RDC between the same asynchronous reset domains

A1. Reset domain crossing with transmitter reset source in different clock – not a safe RDC path

If the transmitter and receiver are driven by the same asynchronous reset domains and the transmitter register is driven by a soft reset from a different clock domain, the change at the input of the soft reset to the transmitter reset can cause an asynchronous assertion at the transmitter, which could cause metastability. Fig. 3a illustrates the RDC issue as an asynchronous reset of Tx flop is driven by a soft reset from a different clock domain. Here, Tx flop and Rx flop are in the rst asynchronous reset domain and TxRstReg is a soft reset for flop Tx. The data of TxRstReg can change in clk clock domain, which can lead to asynchronous assertion of Tx flop in the rx_clk clock domain and result in potential metastability.

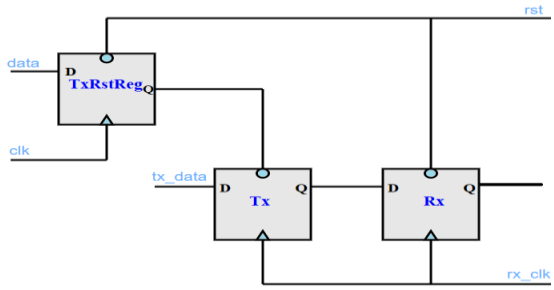

Fig 3a: RDC between Tx flop and Rx flop

A2. Reset domain crossing with transmitter reset source in same clock – a safe RDC path if it meets the static timing analysis requirements

If the transmitter and receiver are driven by the same asynchronous reset domains but the transmitter register is driven by a soft reset that is different than the soft reset of the receiver register or there is no soft reset at the receiving register, it can cause metastability. Fig. 3b illustrates the static timing analysis issue as the path from reset of dff1 flop to the output of dff1 flop is not timed and the soft reset sync1 can cause a setup/hold violation for the dff2 flop and result in potential metastability.

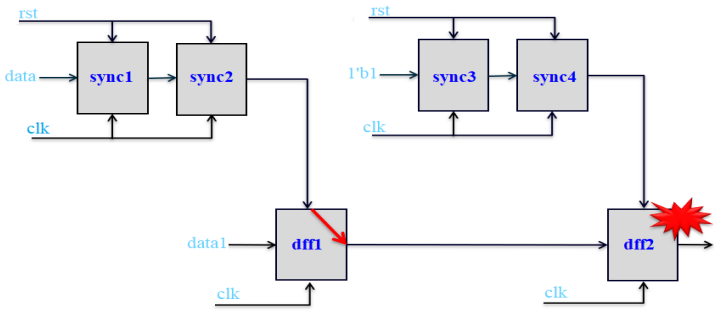

Fig 3b: Safe RDC between dff1 flop and dff2 flop if the static timing analysis requirements are fulfilled

B. RDC between different asynchronous reset domains – safe if it meets the static timing analysis requirements

Now let's look at the case where there is an RDC signal coming from an asynchronous reset source, with a reset in the same clock domain, going to an asynchronous reset destination - If the transmitter and receiver are driven by different asynchronous reset domains, but, the soft reset on the transmitter is generated on the same clock domain as the crossing elements, the change at the input to the destination register when the transmitter reset is asserted is synchronous to the clock, hence there is no potential metastability condition. But, in such

cases, static timing analysis tools are required to be able to time the path from flops controlling the reset net to the D input of the destination flop.

Fig. 3c illustrates the case where, although the asynchronous reset domains are different at the transmitter and receiver, there is no potential RDC. Although, the path from the transmitting to receiving register is safe, as there is no metastability propagating to the Rx flop, we still need to confirm that static timing analysis times the path from the soft reset register TxRstReg through the transmitter flop Tx to the receiver flop Rx. The red highlighted line shows the timing path from flop TxRstReg to the D pin of the destination flop Rx. If the total path delay from the output of the soft register TxRstReg to the Rx data register is less than the clock period minus the setup time of the Rx register, then the metastability issue at the receiver can be avoided. The total path delay in this case can be obtained by adding delays of clock edge to TxRstReg.Q, TxRstReg.Q to Tx.rst, Tx.rst to Tx.Q and Tx.Q to Rx.D. Some static timing analysis tools may time these paths only with options. So, this type of check should be flagged by the RDC analysis tool to alert users that these paths should be timed for setup violations, as, if this path meets the setup time window, then it is safe from metastability related to asynchronously resetting flop Tx.

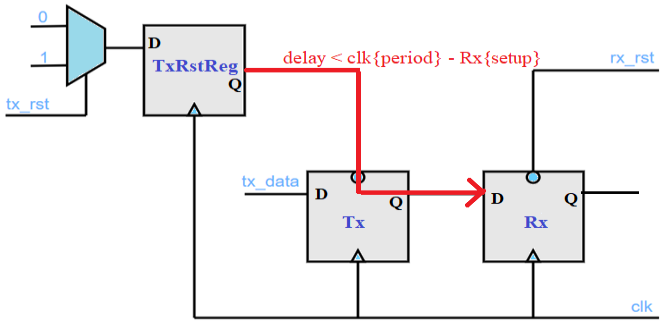

Fig 3c: Safe RDC between Tx flop and Rx flop if static timing analysis requirements are fulfilled

III. PROPOSED METHODOLOGY

With highly complex reset architectures in automotive designs, there arises the need for a proper verification method to detect RDC issues. These RDC bugs, if ignored, can have severe consequences on system functionality, timing, and reliability. It is essential to detect unsafe RDCs systematically and apply appropriate synchronization techniques, and tackle the issues that may arise due to delay in reset paths because of soft resets, to ensure proper operation and avoid the associated risks. By handling RDCs effectively, designers can mitigate potential issues and enhance the overall robustness and performance of a design. As we have already talked about soft resets and related issues, in this section, we propose a systematic flow involving various steps, to assist in RDC verification closure using standard RDC verification tools, see Fig. 4a.

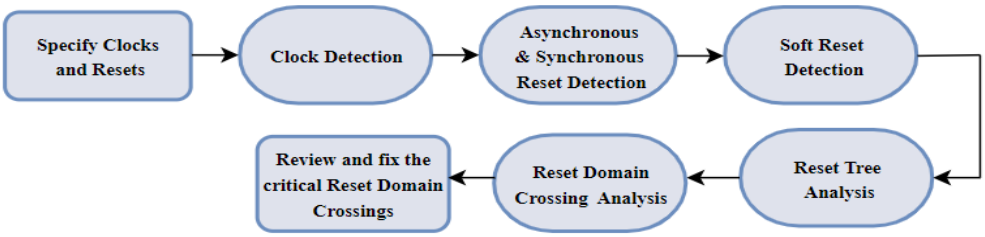

Fig 4a: Flowchart for proposed methodology for RDC verification

A. Specification of clock and reset signals

Signals that are intended to generate a clock and reset pulse should be specified by the user as clock or reset signals, respectively, during the setup step in RDC verification. By specifying signals as clock or reset (according to their expected behavior) designers can perform design rule checks and other verification checks to ensure compliance with clock and reset related guidelines and standards, and best practices. This helps in identifying potential design issues and improving the overall quality of the design by reducing noise in the results.

B. Clock Detection

Ideally, design engineers should define the clock signals and then the verification tool should trace these clocks down to the leaf clocks. Unfortunately, with complex designs, this is not possible as the design might have black boxes that originate clocks or have some combinational logic in the clock signals which do not cover all the clocks that are specified by the user. All the un-specified clocks need to be identified and mapped to the user-specified primary clocks. A proper detection of clocks is required in RDC verification, as potential metastability may occur if resets are used in different clock domains than the sequential element itself and lead to critical bugs.

C. Reset Detection

Ideally, design engineers should define the reset signals but as described above, due to the complexity of designs it is not possible to specify all the reset signals and therefore a proper verification tool is required for detection of resets. All the localized, black-box, gated and primary resets need to be identified, and based on their usage in the RTL they should be classified as synchronous or asynchronous or dual type, and then mapped to the user-specified primary resets. A tree-like structure is formed, illustrating the relationships between various primary resets and localized reset sources (Root in the reset tree is the source reset and the branches are different localized resets)

D. Soft Reset Detection

The soft resets i.e., the internally generated resets by flops/latches need to be systematically detected as they can cause critical metastability issues when used in different clock domains and need static timing analysis when used in the same clock domains. Detecting soft resets helps identify potential metastability problems and allows designers to apply proper techniques for resolving these issues.

E. Reset Tree Analysis

Analysis of reset tree aids designers to identify issues early in the design, before RDC analysis. It helps highlight some important errors in the reset design that are not commonly caught by lint tools; these include:

- Dual synchronicity reset signals i.e., the reset signal with a sample synchronous reset flop and a sample asynchronous reset flop.
- An asynchronous set/reset signal used as data signal can result in incorrect data sampling because the reset state cannot be controlled.
- Signal driven by a tristate logic component or gated by XOR, XNOR, NAND, or NOR used as a reset.
- Dual polarity of reset signals i.e., the reset signal with a sample active low reset flop and a sample active high reset flop.

F. Reset Domain Crossing Analysis

This step involves analyzing a design to determine the logic across various reset domains and identify potential RDCs. The analysis should also identify common reset sequences of asynchronous and soft reset sources at the transmitter and receiver registers of the crossings to avoid detection of false crossings that might appear as potential issues due to complex combinations of reset sources. False crossings are where a

transmitter register and receiver register are asserted simultaneously due to dependencies among the reset assertion sequences, and as a result, mitigates any metastability that might occur on the receiver end. This leads to less noisy results and helps in faster verification closure.

G. Analyze and Fix RDC Issues

The concluding step is to analyze the results of the verification steps to verify if data paths crossing reset domains are safe from metastability. For the identified unsafe RDC's, that may occur due to different asynchronous reset domains at transmitter and receiver end, or due to the soft reset been used in different clock domain than the sequential element itself, the design engineers can develop and implement solutions to eliminate or mitigate metastability by restructuring the design, modifying reset synchronization logic, applying data isolation techniques, or adjusting the reset ordering. For the traditionally safe RDC's i.e., crossings having soft reset used in the same clock domain as the sequential element itself, designer's need to verify the static timing analysis. Refer to Fig. 4b. to identify and eliminate metastability issues due to soft resets. After implementing the RDC solutions, re-verify the design to ensure that the reset domain crossing issues have been effectively addressed.

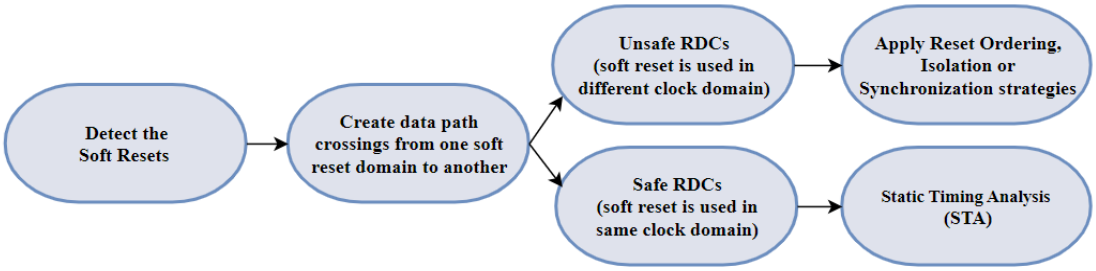

Fig 4b: Flowchart for proposed methodology to tackle metastability issues due to Soft Resets

IV. RESULTS AND SUMMARY

The proposed methodology was used on a design with 374546 register bits, 8 latch bits and 45 RAMs. The Questa® RDC verification tool using the new methodology identified around 131 reset domains in the design, which consisted of 19 asynchronous domains defined by the user as well as 81 asynchronous reset domains inferred by the tool.

A. The first run analyzed data path crossing asynchronous reset domains without any soft reset analysis and reported around 40000 RDC crossings (as shown in Table I).

B. In the second run, we did soft reset analysis and detected 34 soft resets, which resulted in additional violations for RDC paths with transmitter soft sources in different clock domains which are critical and were missed in the initial run. Also, some RDC *violations* were converted to *cautions* (RDC paths with a transmitter soft reset in the same clock domain) as these paths would be safe from metastability if they meet the setup time window (as shown in Table II).

TABLE I
RDC ANALSIS WITHOUT SOFT RESETS

Reset domain crossings without soft reset analysis	Severity	Number of crossings
Reset domain crossing from areset to areset	Violation	28408
Reset domain crossing from areset to non-reset	Violation	11235

TABLE II
RDC ANALYSIS WITH SOFT RESETS

Reset domain crossings with soft reset analysis	Severity	Number of crossings
Reset domain crossing from areset to areset	Violation	26957
Reset domain crossing from areset to non-reset	Violation	10523
Reset domain crossing with tx reset source in different clock	Violation	880
Reset domain crossing from areset to Rx with same clock	Caution	2412

This paper summarizes the need for intelligent reset domain crossing analysis that improves the quality of the RDC verification process by catching violations with the same asynchronous reset domain but different soft resets. It also, reduces the number of unwanted violations, and identifies the cautions that need static timing analysis, to generate accurate results. Validation on real SoCs confirms that the issues due to soft resets are practical and must be taken care of as part of a static RDC methodology to prevent silicon re-spins.

REFERENCES

[1] Akanksha Gupta, Ashish Hari, Anwesha Choudhary, “Systematic Methodology to Solve Reset Challenges in Automotive SoCs”, DVCON Europe 2019
[2] Yossi Mirsky, “Comprehensive and Automated Static Tool Based Strategies for the Detection and Resolution of Reset Domain Crossings”, Intel Corporation
[3] Chris Kwok, Priya Viswanathan, Ping Yeung, “Addressing the Challenges of Reset Verification in SoC Designs”, DVCON US, 2015
[4] Questa® RDC User Guide, version 20.3d, 2021