2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE
MUNICH, GERMANY
DECEMBER 6 - 7, 2022

Reset Your Reset Domain Crossing (RDC) Verification with Machine Learning

Mark Handover, Siemens EDA, Newbury, United Kindom (mark.handover@siemens.com)

***Abstract*— There are many challenges users can face getting started with RDC. These include quickly setting up a baseline for RDC analysis, refining the setup to get more accurate results, understanding related violations to create minimal, accurate constraints, and understanding and managing exceptions. These are challenges well suited to for assistance from machine learning (ML) algorithms. Adding ML assistance to RDC tools provides a big assist to help work through the early issues of getting a design setup and targeting smaller issues as a design matures**

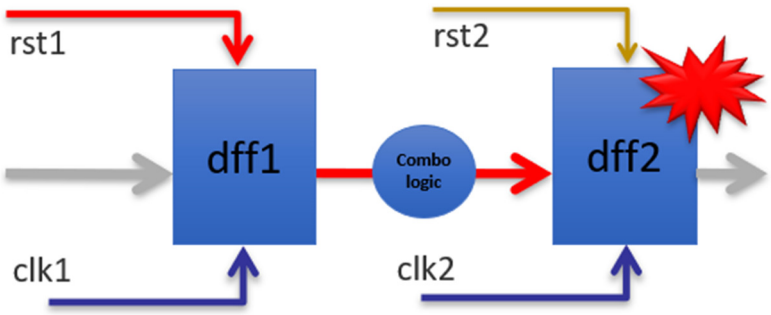
Keywords—RDC, reset domain crossing, EDA, machine learning

I. INTRODUCTION

EDA tools that detect Clock Domain Crossings (CDC) have become a standard quality check in today's design space. A bit newer to the game is the concept of Reset Domain Crossing (RDC) verification. Improper reset domain handling can cause the same metastability issues, and unpredictable behavior as clock domain crossing errors. Consider the example in figure 1.

If rst1 is asserted while rst2 is not asserted, the asynchronously changing reset value of dff1 may change close to the edge of clk2, causing metastable values to be sampled in dff2. However, there is no issue if rst2 is already asserted when rst1 is asserted. CDC tools can verify issues where the clocks and resets are both asynchronous. If the clocks are synchronous, evaluation by an RDC tool would result in a violation being reported if reset ordering constraints have not been applied properly.

Figure 1



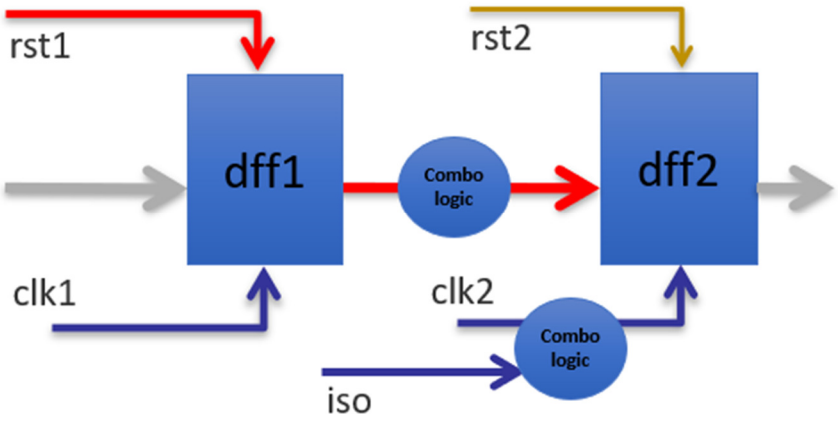
II. THE SETUP AND NOISE REDUCTION CHALLENGE

As CDC and RDC are inextricably linked, the challenges inherent in setting up RDC for successful verification are predicated by use of CDC analysis as a starting point. Many of the setup directives and constraints can be inherited from the CDC tool for defining clocks and synchronous clock groupings, specifying constants, pseudo-static signals, modes of operation, custom synchronization schemes etc. But starting RDC analysis requires further constraints and setup; constraints to describe the sequencing of asserting or de-asserting resets, ties to the clocking architecture, clock, and data isolation issues, as well as handling 3rd party IP's with unique reset requirements are just a few of the areas that must be addressed.

Consider the earlier example. RDC analysis needs to understand whether rst1 and rst2 are asynchronous, but also whether clk1 and clk2 are synchronous or asynchronous as this will impact the RDC analysis. There may be further constraints specified by the user, for example to specify that rst2 always asserts before rst1. This is classed as a reset ordering constraint. Consider now figure 2.

In this scenario we have a similar RDC crossing, but the RX clock is gated with the isolation signal, "iso." This is a common RDC strategy. During the reset sequence, the isolation signal disables the RX clock, preventing any data from being sampled into dff2 which stops metastability being sampled and propagated. RDC tools need to be guided regarding which signals should be considered as isolation gating signals during RDC analysis before further in-depth verification of the safety of the RDC crossing is performed. These constraints are classed as RDC clock isolation constraints. There are other schemes with additional constraints that can be specified to provide guidance to the RDC tool for allowing suitable analysis and classification of the RDC crossings.
This added complexity, layered on top of the CDC baseline, has made setting up the constraints properly and getting actionable information from the tool a challenge in the early phases of development.

Figure 2



Consider a multi-million gate design with dozens of resets and initially 10's of thousands of RDC violations. Manually reviewing the RDC crossings and defining the reset architecture constraints to the RDC tool can take weeks. Which crossings require reset ordering constraints? Which crossings require some form of isolation constraint? Are the resets and clocks fully and relationships correctly defined? Are there missing setup constraints for modes or constants? There are multiple challenges users can face with RDC, including:

Quickly setting up a baseline of RDC analysis
Refining the setup to get more accurate results
Understanding related violations to create minimal, accurate constraints
Understanding and managing exceptions

Fortunately, these are problems that are well suited for assistance from Machine Learning (ML) algorithms. Adding ML assistance to RDC tools is an accelerator to help work through the early issues of getting the tool setup. ML also has the ability to target smaller issues as maturity increases. When first starting with an RDC tool, the ML algorithm can identify resets, including gated resets, as well as provide the designer a list of suggested commands and constraints that can be cut and pasted into their scripts to improve the quality of results the tool is reporting. The algorithms also provide a mechanism for the designer to add thresholds to influence the analysis. By following a methodology where the algorithms only report the largest RDC violators, designers can address the large problems first because the tool has filtered out a lot of the low level "noise" from the report for them. As constraint definition matures, these thresholds can be iteratively lowered until the design is clean.

III. STUDY

This paper will examine real world data to assess the benefits of applying machine learning assistance to a design. The reference design had already been run through RDC using manually generated constraints, making it an ideal case study for the accuracy and effectiveness of machine learning assistance. We will compare the performance impact, amount of noise, and the results to determine how close the machine learning came to the hand generated constraints.

The chosen reference design had over 10 million state bits, 3 clock groups, 8 reset groups and 90k RDC paths that included the need for reset gating, reset ordering, and clock isolation.

The machine learning assistance flow in Questa RDC suggests constraints that the user can then review and apply. To compare the usefulness of these constraints against the known good manual constraints, we required a baseline design with minimal RDC setup to which we could apply the ML suggested constraints, thus allowing a comparison of the RDC results to those from the original manual setup. To set the baseline, the design was run through Questa RDC with no waivers, no RDC ordering constraints and no RDC isolation constraints, which resulted in over 10,000 RDC violations for paths between different asynchronous reset domains.

The assist flow generates a report which details several constraint suggestions, along with the number of RDC paths impacted by that constraint suggestion, an example of which is shown in Figure 3.

Figure 3

```
============================================================
Section 1 : Possible reset order
============================================================

1. fifo_1_d.u_rafifo.u1.rst0
   u_csr_interface_apb.u_samplefifo.rp_gray.rst_gbl
     Affected crossings (150)
        rdc_areset_209679,  rdc_areset_217871,  rdc_areset_193295,  rdc_areset_201487...(146 more)

     Constraint :
        resetcheck order assert -from {fifo_1_d.u_rafifo.u1.rst0} -to {u_csr_interface_apb.u_samplefifo.rp_gray.rst_gbl}

============================================================
Section 2 : Possible resets with same domain
============================================================


1. u_csr_interface_apb.u_samplefifo.rp_gray.rst_gbl
   u_master_interface.u_apb_master_mc.rst_m
     Affected crossings (100)
        rdc_areset_10027,  rdc_areset_1515,  rdc_areset_1522859,  rdc_areset_1531115...(96 more)

     Constraint :
        netlist reset {u_csr_interface_apb.u_samplefifo.rp_gray.rst_gbl} -group <group name>
        netlist reset {u_master_interface.u_apb_master_mc.rst_m} -group <group name>
```

The first pass of ML RDC assist reported the results in table 1.

Table 1

Suggested Constraint	Number of unique suggestions	Number of RDC paths affected
Reset ordering	1	Greater than 3000
Reset ordering	3	Greater than 100
Reset ordering	5	Less than 100
Common Reset Domain	1	
RDC clock isolation signal	4	Greater than 3000
RDC clock isolation signal	3	Less than 600
RDC clock isolation signal	7	Less than 100

From the results we see that, for this design, there are 3 types of constraint suggested by the assist flow:

Reset ordering
Reset grouping
Clock isolation signals

For each type of constraint, various suggestions were made. For example, in one case the reset ordering suggestion might be from RstA to RstB. In another, the suggestion might be from Rstx to RstY. In each case, the constraint will impact multiple RDC crossings, and these are also listed in the report such that the user can focus initially on constraints that may have the largest impact.

As part of the assist flow, the user may use a threshold parameter to allow RDC assist only to report constraints which impact a total number of RDC paths above this threshold. As part of our experiments a new analysis was performed where the threshold was set to a minimum of 500 paths globally, and this was reflected in new constraint suggestions, table 2.

Table 2

Suggested Constraint	Number of unique suggestions	Number of RDC paths affected
Reset ordering	1	Greater than 500
Common Reset Domain	1	
RDC clock isolation signal	5	Greater than 500

A brief manual comparison of the reset ordering constraints confirmed that they mapped closely to the original hand-crafted user constraints.
After a review of the common reset domain constraints, it was determined that this suggestion was inappropriate for this design configuration. The constraint was sensible, but not applicable for the architecture in this case.
Applying this suggested set of unmodified reset ordering and clock isolation constraints to our baseline setup dropped the number of RDC violations from over 10000 to 440.

From what we hear when working with customers running RDC checks, reset ordering constraints, while they can be challenging, are often better understood, and can typically be more clearly defined in architecture specifications than other required constraints. The results in table 3 show the number of ordered RDC paths from the last run, compared to the original RDC results which used the hand-crafted constraints.

Table 3

	Baseline	Hand-written constraints	RDC Assist constraints
# Of Ordered Reset domain crossings reported after RDC run	0	6185	3532

This data shows that whilst the reset ordering constraints from the assist flow were having a significant impact on our results, resolving 3532 RDC crossing violations, they were not complete when comparing with the hand-crafted constraints (not withstanding that the assist constraints were limited due to the application of the 500-path threshold).

In keeping with our observation from customers that isolation constraints are far more difficult to specify than ordered reset constraints, our next step was to run RDC analysis with a mix of user and RDC assist constraints. i.e., apply all the user reset ordering constraints alongside the 14 clock isolation constraints from the first ML assist pass. The results are shown in table 4.

Table 4

	Baseline	Hand-written constraints	RDC Assist constraints
# Of isolated clock gated RDC crossings	0	4446	4436

The results proved that the constraints from the RDC assist flow were highly accurate, constraining to within just 10, the number of isolated RDC paths compared to the design with manual constraints.
Applying this suggested set of unmodified clock isolation constraints to our baseline setup, along with the user ordered assert directives dropped the number of RDC violations from the initial 10000+ (in the unconstrained baseline design) to a total of just 221 violations.

IV. CONCLUSION

Comparing the final ML results against the original, with manually generated constraints reveals matching results across most of the RDC evaluation criteria, with a small difference in just a handful of crossings. There was no noticeable runtime or memory impact when enabling the RDC assist capability, compared to the baseline run.

These experiments have proven that RDC ML assist is an incredibly useful technology in helping users converge more quickly on constraints, easing setup overhead and quickly reducing the noise in RDC results. It is an easy to use, minimal impact method which makes it simple to adopt and provide value. Anecdotal reports of the effort in converging on accurate RDC constraints often speak of weeks or even months. Questa RDC assist provides useful, accurate and actionable results with minimal impact to the current RDC flow or runtimes enabling customers to vastly reduce their time to RDC closure.
The usage of mixing auto-generated and user constraints reflects the more typical use case expected when using the RDC ML assist flow. i.e., the user will have setup up a level of baseline constraints, then enable the ML flow

to help with RDC setup refinement and guidance to more quickly complete RDC verification. However, this ML assist technology would equally serve those customers wishing to adopt this to simply act as a helpful guide during constraint reviews and architecture intent discussions when working to close out the RDC verification task.

REFERENCES

[1] DVCon Europe, Oct 2021: Resetting RDC Expectations: A Systematic Approach to Verifying Configurable Designs, and Detection of glitch-prone clock and reset propagation with automated formal analysis

[2] DVCon China, May 2021: Build Reliable and Efficient Networks with a Comprehensive Reset Domain Crossing Verification Solution

[3] DVCon U.S, March 2021: Handling Reset Domain Crossing for Designs with Set Reset Priority Flops, and Bringing Reset Domains and Power Domains Together – Confronting Issues Due to UPF Instrumentation

[4] DVCon Europe, October 2020: Build Reliable and Efficient Networks with a Comprehensive Reset Domain Crossing Verification Solution, and Achieving Faster Reset Verification Closure with Intelligent Reset Domain Crossings Detection