Addressing Reset Domain Crossing Issues at Block-Level and Top-Level using VC Spyglass in Complex SoC

Isuru Athapattu, Harshavardhan Vajjaramatti

Nokia Solutions and Networks Oy, Finland

www.nokia.com

ABSTRACT

'Reset' is one of the most fundamental aspects of semiconductor development. From the initialization of entire hardware designs to clearing of all the software running through the SoC and restoring the device to a known state, resets serve their purpose. The current trend towards utilizing low-power techniques and incorporating heterogeneous functionalities into a single SoC core has led to the requirement of designing complex reset structures with multiple asynchronous reset domains. Based on current growth and future trends, the SoCs are becoming even more complex. Due to this, the number of reset domains required might grow exponentially, resulting in the need to handle an extremely large number of Reset Domain Crossings (RDC), which can cause chip failures if not taken care of.

Resolving RDC issues requires the implementation of various types of blocking and synchronization mechanisms in the design, while performing a thorough static analysis in a proper EDA tool environment. This paper discusses the design principles and RDC blocking/synchronization techniques and how VC SpyGlass (VCSG) can be effectively used to resolve the RDC design issues including clock glitches, datapath glitches, and convergence of RDC paths. In addition, VCSG also helps RDC analysis by skipping reset-less flops. The hierarchical RDC verification is a very useful and effective flow, which will also be discussed in the context of SoC verification sign-off.

Table of Contents

1. Introduction 5
1.1 Metastability 5
1.2 Types of Resets in SoC 5
1.2.1 COLD Power-on-Reset (PoR) 6
1.2.2 WARM Power-on-Reset (PoR) 6
1.2.3 Hardware(HW) Resets 6
1.2.4 Software(SW) Resets 6
1.3 Issues Related to Resets 7
1.3.1 Reset Architecture 7
1.3.2 Reset Utilization 7
1.4 Reset Domain Crossings 7
2. Challenges in RDC Verification 9
2.1 Block-Level RDC Verification 9
2.1.1 Data Corruptions 9
2.1.2 Clock Corruptions 9
2.1.3 Glitches 9
2.1.4 Convergence 10
2.2 Top-level RDC Verification 10
2.2.1 Migrating block-level constraints to the top-level 10
2.2.2 Violation count 10
2.2.3 Usage of resources 10
3. Addressing RDC issues 10
3.1 RDC Techniques 10
3.1.1 Clock Gating 10
3.1.2 Data Gating 12
3.1.3 Flop Enable/Disable 12
3.1.4 Reset Re-sequencing 13
3.1.5 Flip-flop Synchronizers 14
4. Proposed RDC Verification Methodology 16
4.1 Block-level RDC Verification 16
4.1.1 RDC resolved by reset assertion re-ordering. 16
4.1.2 RDC with resetless flop chain handled with Clock-Gating 16
4.1.3 RDC handled with Data-gating 17
4.1.4 Clock Corruptions 18

4.1.5 RDC Glitch 18
4.1.6 RDC Convergence 19
4.2 Top-level RDC Verification 20
4.2.1 Design under consideration 20
5. Results 21
5.1 RDC Analysis and resolution of IP1 21
5.1.1 Solution 1 : Fixing Constraints 21
5.1.2 Solution 2 : Clock Gating 21
5.2 RDC Analysis of top in hierarchical flows 22
6. Conclusions 22
7. References 23

Table of Figures

Figure 1 : (a) D Flip-flop (b) Metastability window and waveform 5
Figure 2 : (a) Hierarchy of Reset Impact (b) General reset signal distribution. 6
Figure 3 : Reset Domin Crossing basic scenario. 8
Figure 4 : RDC Waveform / Data Corruption due to asynchronous reset assertion at the source 8
Figure 5 : Clock corruption due to asynchronous reset assertion. 9
Figure 6 : Clock gating mechanism. 11
Figure 7 : Data gating mechanism. 12
Figure 8 : Flop Enable/Disable mechanism. 13
Figure 9 : Initial Reset Assertion Hierarchy 14
Figure 10 : Altered Reset Assertion Hierarchy 14
Figure 11 : Flip-flop Synchronizers 15
Figure 12 : Occurrence of uncertain one clock cycle delay across a flip-flop synchronizer 15
Figure 13 : RDC fixed by reset assertion order. 16
Figure 14 : RDC through resetless flip-flop chain. 17
Figure 15 : RDC fixed by Clock-gating. 17
Figure 16 : RDC fixed by Data-gating. 17
Figure 17 : Clock corruption at CGC due to asynchronous reset assertion. 18
Figure 18 : RDC paths combined before destination. 18
Figure 19 : Combined RDC paths detected as glitch source to destination flip-flop. 19
Figure 20 : Glitch potentially blocked by a data gate qualifier. 19
Figure 21 : Separately synchronized RDC paths converges at destination. 19
Figure 22 : Top level view of the DUT 20

Table of Tables

Table 1: Design Complexity Summary 20
Table 2: IP1: Violation count summary. 21
Table 3: TOP: Violation summary 22

1. Introduction

In the realm of digital design, where intricate systems are meticulously crafted to perform complex tasks, ensuring the reliability and robustness of these systems is paramount. One critical aspect of this assurance lies in the management of reset signals as they traverse through different clock domains within the design. It's crucial to establish a foundational understanding of the overarching principles governing digital systems and the role of resets within them. Digital systems, characterized by their discrete states and operations, rely on clock signals to orchestrate synchronous behavior. However, as designs grow in complexity, incorporating multiple reset domains becomes inevitable to meet performance and power requirements. There is a substantial significance for resets in digital circuits, serving as the mechanism to initialize or reinitialize the system to a known state. Yet, when these reset signals are asserted partially creating multiple reset domains, challenges arise. As such, the introductory chapter delves into the fundamental concepts of metastability, various types of resets in a complex SoC and Reset Domain Crossings (RDC). It lays the groundwork for subsequent discussions on the intricacies of reset domain crossings, addressing the complexities, challenges, and methodologies employed to mitigate risks and ensure the reliable operation of modern digital designs.

1.1 Metastability

Metastability is the phenomenon in digital circuits where a sequential element such as a flip-flop fails to resolve to a stable logic state of either 0 or 1 within a predefined time. The root cause for this occurrence is the violation of setup-time or hold-time requirement of the flip-flop. i.e., when a data transition is observed at the input, overlapping with the critical metastability window of the active edge of the operating clock as shown in Figure 1. This can occur whenever,

- The input is a signal from a different asynchronous clock domain (CDC).
- The clock rise and fall transition times are higher than the tolerable limit.
- The source and destination frequencies are equal, but have different phases or skews.
- Combinational delays in the input data path such that the state transition overlaps the metastability window. (Negative Slack)
- The input signal from same clock domain changes upon an asynchronous reset assertion at the source (RDC).

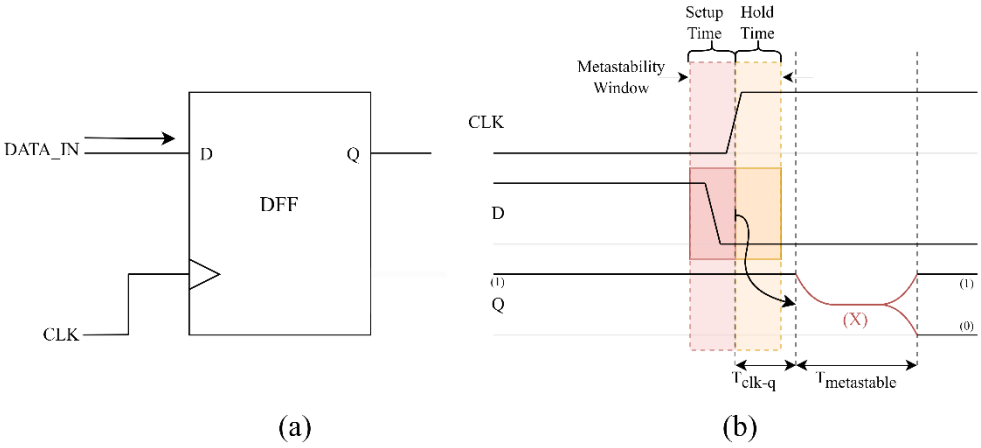


Figure 1 : (a) D Flip-flop (b) Metastability window and waveform

1.2 Types of Resets in SoC

Different types of reset domains can be identified according to the scope of impact or the depth of propagation on the chip. These resets are triggered by various stimuli and each type is responsible for performing specific tasks related to system initialization or recovery in different levels of hierarchy

in the chip.

1.2.1 COLD Power-on-Reset (PoR)

Cold power-on-reset is done when the chip is completely powered off, and the power is then reapplied to initialize the system. Typically, it is triggered by an external signal, which in certain designs, the triggering occurs internally upon encountering under voltage conditions. It affects the entire chip, essentially resetting every circuit, processor, memory element, and interface. And cold PoR has the deepest propagation, as the reset assertion affects the entire chip, affecting every module hierarchy from SoC top to leaf cell level.

1.2.2 WARM Power-on-Reset (PoR)

Warm power-on-reset is performed when the chip is already operational, and a restart is initiated without completely taking away the power. This is usually triggered by an external signal and in some cases triggered internally during a self-system recovery. It is often used to reset the entire SoC or specific subsystems, clearing most operational states, but it may retain some settings or states. POST(Power-On-Self-Tests) may not be executed during a Warm PoR. The impacting scope and the propagation depth of the warm PoR reset within the chip are less than those of cold PoR reset.

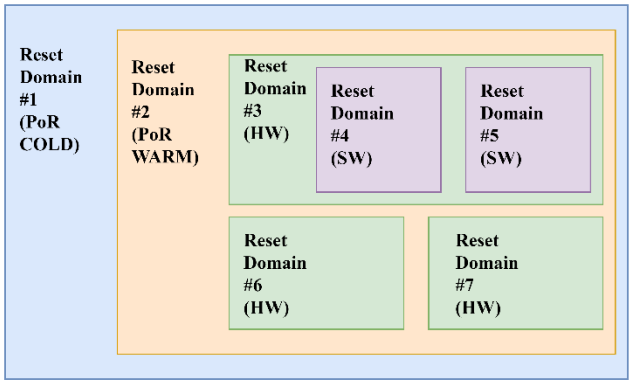
1.2.3 Hardware(HW) Resets

These resets are automatically triggered by built-in hardware mechanisms in response to certain events/conditions. It is more localized than PoR reset, affecting only designated partitions or IP blocks within the design. Hardware resets are usually internally generated in the reset controller unit of the SoC.

1.2.4 Software(SW) Resets

Software-triggered resets have a more limited impact on the system-on-chip (SoC) compared to hardware resets. This type of reset does not involve any physical actions or external signals. They are used to clear or reinitialize software-driven states or data paths without impacting the hardware configuration. For instance, a watchdog timer serves as a trigger for this type of reset. The attempt aims to restore normal operation without causing disruptions to other functions of the SoC.

Figure 2. (a) depicts the hierarchy of the reset domain types and (b) is tabulated to elaborate the triggering of reset signals which are distributed throughout the chip upon assertion of each type of reset domain. In summary, the assertion of a certain domain forces the resets in a subdomain to assert as well. For instance, *rstn_1* in SW reset domain #4 undergoes assertion upon the assertion of HW reset domain #3.



(a)

Reset Signal	Reset Domains					
	RD#1 (PoR COLD)	RD#2 (PoR WARM)	RD#3 (HW)	RD#4 (SW)	RD#5 (SW)	RD#6 (HW)
rstn_1	X	X	X	X		
rstn_2	X	X	X			
rstn_3	X	X	X		X	
rstn_4	X	X				X
rstn_5	X	X				X
.	X	X				
rstn_k	X					

(b)

Figure 2 : (a) Hierarchy of Reset Impact, (b) General reset signal distribution.

Apart from the aforementioned reset types, there are special resets which are used during the testing

and debugging phases of SoC development and maintenance. They allow developers to reset certain parts of the system without affecting the entire SoC, facilitating targeted testing and debugging.

1.3 Issues Related to Resets

There are two principal categories of issues related to resets: Improper reset distribution tree and reset utilization.

1.3.1 Reset Architecture

This refers to the problems that arise from the architecture of how reset signals are sent across the circuit.

- Using a reset signal in both asynchronous and synchronous designs can cause timing conflicts.
- Misuse of reset lines for data or control signals can introduce functional errors.
- Having flip-flops with simultaneous set and reset conditions can lead to indeterminate states.
- Resets with misaligned timings for activation and deactivation can disrupt proper signal propagation.
- Inefficient reset structures which are unnecessarily complex.

1.3.2 Reset Utilization

This refers to the problems that arise from the architecture of how reset signals are sent across the circuit.

- Failing to synchronize an asynchronous reset upon de-assertion can result in metastability.
- Incorrect polarity and clock usage after synchronization.
- Challenges with Reset Domain Crossings (RDC) during assertion, which can lead to timing violations and data corruption.

1.4 Reset Domain Crossings

A "reset domain" refers to a specific partition of the chip that operates obeying a unique reset signal or a set of reset signals which, by logic or by convention, get asserted synchronously and simultaneously during the cycle operation. A reset domain can be identified as either of the types in Section 1.2 depending on their scope of impact. This division, which can be defined with respect to a logical or physical boundary, guarantees that certain segments of the SoC can undergo a reset independently without impacting the entire system.

Similarly to clock domain crossings, data and control signals upon crossing reset domains can introduce a potential threat to the integrity of the whole system if not handled properly. Asynchronous resets upon their assertion can create a situation which can violate the setup or hold time requirements of the flip-flops, and eventually propagate a corrupted signal.

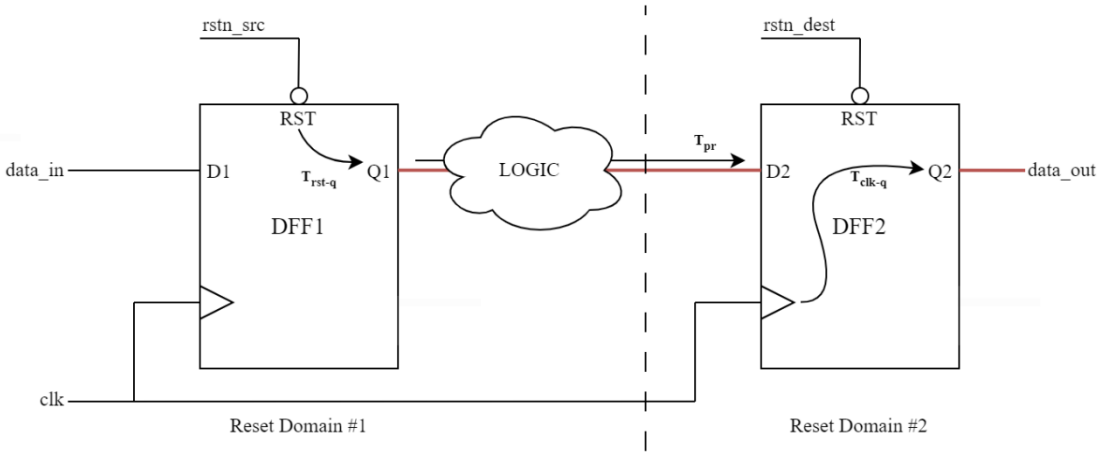


Figure 3 : Reset Domin Crossing basic scenario.

In Figure 3, suppose that the source reset *rstn_src* is asynchronous with the destination reset *rstn_dest*. If *rstn_src* is asserted while the destination reset *rstn_dest* is still in function, this can cause the output of the flip-flop DFF1 to change asynchronously with the potential threat of overlapping with the metastability window of the flop DFF2 as shown in Figure 4.

Metastable states are possible in all the following scenarios.

- The source and destination clocks are from two asynchronous clock domains.
- The source and destination clocks are synchronous with each other, or it is the same clock.
- The synchronicity between the transmitting and receiving clock domains does not affect the occurrence of RDC metastability.
- De-assertion of the source reset synchronous to source clock and the the destination reset synchronous to destination clock. This only ensures that the individual flip-flop outputs do not go to a metastable state during the de-assertion of their resets.
- The reset of the destination is synchronous with the destination clock domain.
- The destination flip-flop is resetless.

The list of conditions mentioned above can be narrowed down to the sole factor that is considered in the context of Reset Domain Crossings, which is the assertion of the reset of the source flip-flop, asynchronous to the active edge of the operating clock at the destination flip-flop.

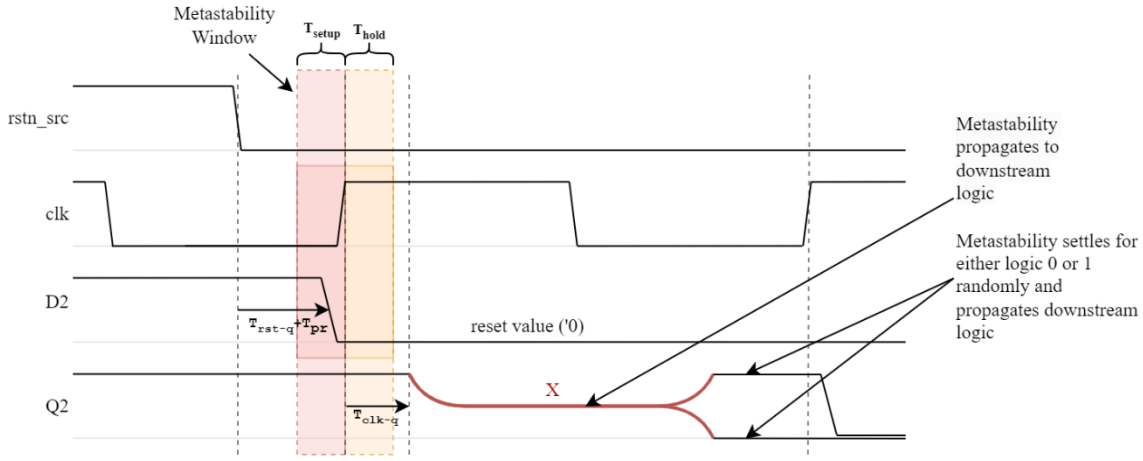


Figure 4 : RDC Waveform / Data Corruption due to asynchronous reset assertion at the source

2. Challenges in RDC Verification

RDC verification is a crucial aspect in the digital design flow aimed at ensuring the proper functionality and robustness of a system when the reset crosses boundaries. The design flow of System-on-Chip involves a complex hierarchy, which includes the integration of different subsystems such as processors, memory units, and interfaces onto a single chip core. To maintain structural and functional integrity throughout the design process, from the initial stages of RTL coding, static verification of the RTL is performed. At the block level, RDC verification focuses on analyzing and validating the safety of signal interactions between different reset domains within a single subsystem or block. And at the top-level, RDC verification is intended to address RDC issues arising from the interactions between the subsystems postintegration at the SoC top level. Specific methodologies are employed to streamline these two processes simultaneously to effectively manage the project timeline and the complexity of SoC design. It is crucial to have a thorough understanding of the challenges associated with each of these processes.

2.1 Block-Level RDC Verification

2.1.1 Data Corruptions

The logic transition in a data path between two reset domains can become asynchronous to the destination clock if the source reset is asserted asynchronously to the destination clock domain. This may create data corruption in the destination reset domain if the input data transition overlaps with the metastability window of the destination clock. This is shown in Figure 4

2.1.2 Clock Corruptions

Clock-gating techniques are commonly employed in chip cores to meet lower-power requirements and block RDC paths. Typically, clock enable signals are produced as registered outputs and utilized directly in the clock gating cell when the source and destination clock domains are the same. However, if they differ, a synchronizing cell must be utilized to synchronize the clock enable signal to the destination clock domain. In both scenarios, the clock enable signal may become asynchronous with the destination clock domain due to an asynchronous reset assertion at the output register or the synchronizing cell. As depicted in Figure 5, if the enable signal at the input of the CGC overlaps with the metastability window of the clock that needs to be gated, it can lead to metastability in the clock network post-gating. This can result in the malfunctioning of all sequential elements that are driven by this gated clock.

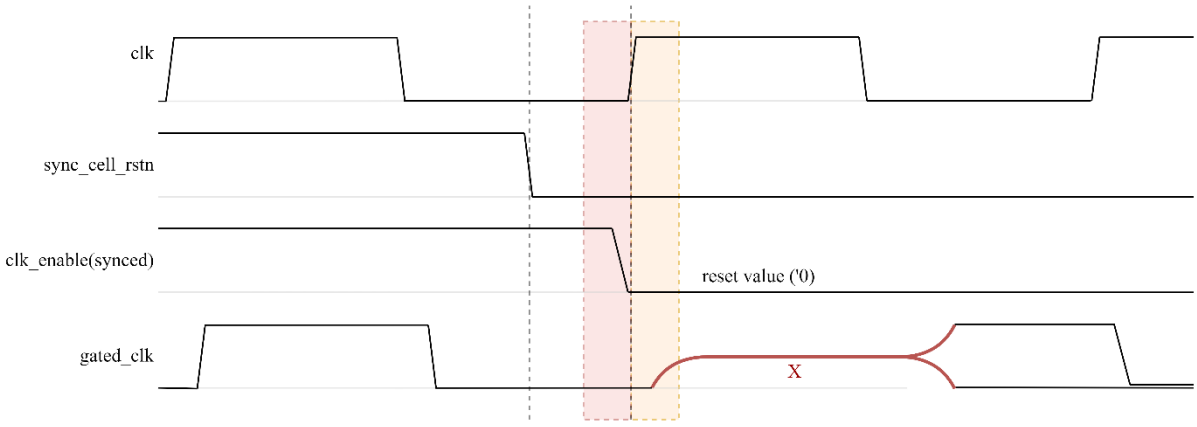


Figure 5 : Clock corruption due to asynchronous reset assertion.

2.1.3 Glitches

The combination of multiple paths with different propagation delays from the same or different reset

domains can result in glitches upon the assertion of resets at the sources. Occurrence of glitches could lead to relaxing the reliability of flip-flop synchronizers as the Mean Time Between Failures (MTBF) of the flip-flop synchronizer is adversely affected.

2.1.4 Convergence

Convergence issues may arise when several signals transition from one reset domain to another domain, which may feature a different reset configuration or no reset at all yet remain independently synchronized. Utilizing RDC converged resultant signal could lead the device to unknown states as functionally invalid signal values could propagate upon convergence.

All above RDC issues can be detected using VC SpyGlass (VCSG) with proper configurations and necessary design constraints. We will discuss more about VCSG usage in sections 4 and 5.

2.2 Top-level RDC Verification

During top-level RDC, in addition to the challenges or the issues caught in block-level, several other complications arise.

2.2.1 Migrating block-level constraints to the top-level

For a thorough and noise-free Static RDC analysis, it is essential to constrain the design with 100% accuracy at the block-level. Manually migrating these low-level constraints to the top level is an exhaustive task in an extremely complex design such as an SoC.

2.2.2 Violation count

At the top level, an SoC integrates numerous blocks, each with its own reset domain. The interactions between these blocks across different reset domains significantly increase the potential for violations, as signals cross between domains more frequently and in more complex ways than within individual blocks. Even small inaccuracies or minor issues in block-level reset domain management can accumulate and manifest as violations at the top level when blocks are integrated, due to the interdependencies and cumulative effects of these issues.

At the top level, analyzing the internal logic of the subsystems or blocks for RDC may result in the accumulation of block-level violations at the top, which is noisy. A hierarchical RDC sign-off is done at the top level utilizing Sign off Abstract Models (SAM). Usage of these SAM models can resolve all these issues significantly.

2.2.3 Usage of resources

The synthesis and structural analysis of the top-level design are more demanding in terms of processing power, memory, and disk space compared to any of the block-level subsystems. This is because the complexity of the design reaches its peak at the top level. The license usage time could also be longer due to the higher runtime and sign-off time needed, as addressing and debugging all the issues at the top level require collaboration among multiple teams, and the number of violations could be significantly high.

3. Addressing RDC issues

Addressing Reset Domain Crossings issues involves implementing strategies to mitigate potential hazards and ensuring the proper functioning of digital systems when the data/control path crosses reset domains. Below are some approaches.

3.1 RDC Techniques

3.1.1 Clock Gating

Clock gating is a widely used technique for managing Reset Domain Crossings in SoCs. Its primary

function in this context is to control when a flip-flop is allowed to capture data. Clock gating acts to shut down the destination domain capturing flip-flop and prevent any asynchronous signal changes from being captured and propagates to the downstream logic. Figure 6 illustrates how a Clock Gating Cell (CGC) is used to control the clock input of DFF2. The minimum requirement to resolve the RDC issue of this implementation is to avoid one active clock edge near the source reset assertion.

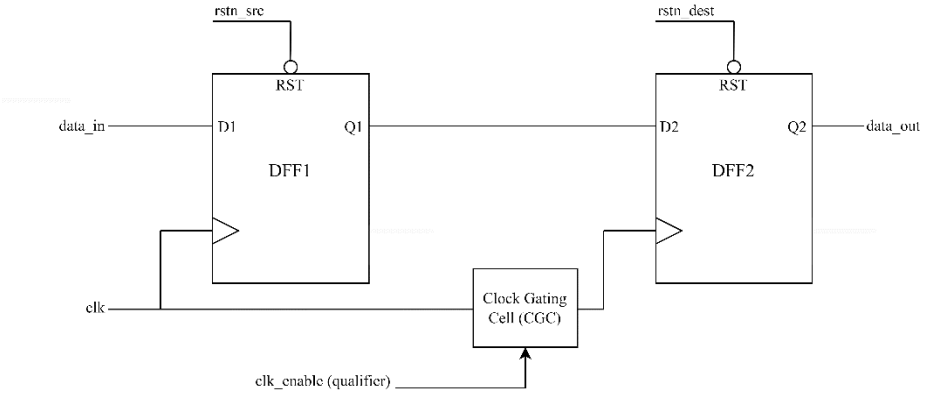


Figure 6 : Clock gating mechanism.

The CGC requires an extra control signal *clk_enable*, which is also known as the qualifier signal to initiate the gating action. The behavior of the qualifier needs to be implemented in a separate module such as a reset generation unit, and it needs to be synchronized to the destination clock domain in the case of CDC. When a reset is required in the source domain, before asserting that reset, the clock enable signal needs to be lowered so that it blocks the clock, and the destination flip-flop cannot receive an active clock edge to initiate the capturing process. When the clock is gated successfully, the source reset can be asserted safely. The clock revival can be implemented in two ways. If the reset value of the source needs to be captured and propagated downstream immediately, the clock needs to be reactivated one clock cycle after the reset assertion. Otherwise, if the last value stored in the capture flop needs to be held until the source starts functioning again, the clock needs to be revived after de-asserting the source reset. Using clock gating for reset domain crossings (RDC) offers several advantages in digital circuit design, particularly in complex systems like SoCs.

- In normal operation, no additional propagation delay is added to the data path. This is beneficial for optimizing the performance of the system with respect to the clock.
- A single CGC that is capable of handling high fan-out can be used to block all RDC between two domains. Each RDC path does not need to be handled separately.
- Reduced dynamic power consumption due to the disabled clock during reset assertion, which prevents flip-flops from switching unnecessarily and can be revived only when required.

While clock gating is a useful technique for managing Reset Domain Crossings (RDC), it does have some potential disadvantages.

- An additional qualifier signal is required to be generated for clock gating. This increases the complexity of the design and the verification workload, as additional checks are required to ensure the accurate behavior of the newly added qualifier logic.
- If the qualifier signal is generated from a different clock domain, then it needs to be synchronized to the destination clock domain, and this requires an additional synchronization scheme.
- Increases the complexity of the clock distribution tree and adds additional gate delays to the clock path. Therefore, it needs to be handled properly in clock tree synthesis.

In most of the industrial cell libraries, clock gating cells are included and can be utilized without writing a separate RTL code.

3.1.2 Data Gating

Data gating is a commonly employed method for handling Reset Domain Crossings in System-on-Chips. In contrast to clock gating, which blocks the clock, data gating is intended to block the asynchronous data path before it reaches the data input pin of the destination flip-flop. This avoids the signal transition from overlapping with the active edge of the destination clock, which as a result vanishes the possibility of metastability. Figure 7 illustrates a simple scenario in which data gating logic is used. The minimum requirement to resolve the RDC issue of this implementation is to delay the data transition upon an asynchronous reset assertion from reaching the destination flop for at least one active edge of the destination clock.

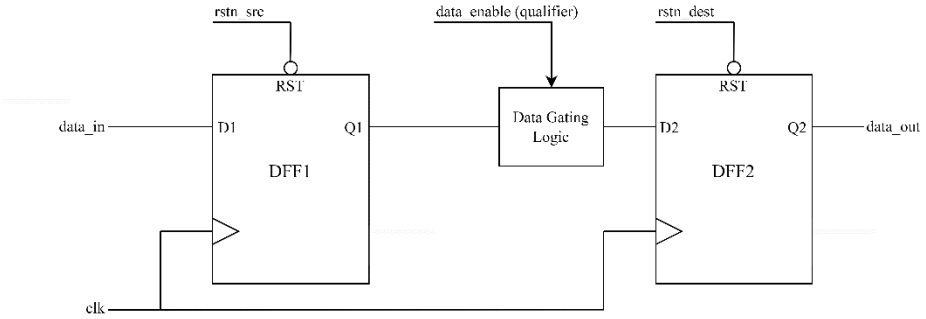


Figure 7 : Data gating mechanism.

Here, the data blocking scheme usually refers to a simple AND or OR gate with a qualifier signal and data as input. AND gate is used when the destination needs to capture logic level 0, and the OR gate if it is 1 upon source reset. This depends on the set/reset configuration pair of the source and destination flops. Similarly to clock gating schemes, data gating also requires a qualifier signal *data_enable* as shown in Figure [REF] that must be synchronized with the destination clock domain. The data-enable signal needs to be altered such that data propagation is blocked before the source reset is asserted.

Data gating for Reset Domain Crossings offers a different set of advantages over other methods.

- Data gating focuses on controlling the data paths instead of clock, which can lead to a simpler clock network since the need for complex clock gating logic is reduced, potentially easing the clock distribution and timing closure process.
- The blocking scheme is simpler and easier to implement.

Data Gating inherits some potential disadvantages as well.

- An additional qualifier signal is required to be generated for data-gating between two reset domains, and this signal must be synchronized with the destination clock domain, which requires an additional synchronization scheme. Similarly, as clock-gating, additional verification needs to be carried out to ensure the qualifier behavior.
- Implementing a gating scheme for all crossings is required separately, unlike clock gating, and each path is introduced with an extra combinational delay, which affects the design timing optimizations.
- Might increase the power utilization due to added logic, and the clock switching is continued to function.

3.1.3 Flop Enable/Disable

This technique is like data gating. Here, instead of a pure logic gate, a 2X1 MUX is used to select the

data path from the source domain or the tied constant. Most cell libraries include a cell that is a combination of a D flip-flop and an input MUX. Using an already exciting D flip-flop library cell with an enable pin may avoid the need to implement a separate gating mechanism at the top level. As shown in Figure 8, the *flop_enable* qualifier is connected to the EN port, which acts as the select signal of the MUX inside DFF2. The advantages and disadvantages of the data-gating scheme are common to this technique, as well expected for the aforementioned. However, there might be an increase in propagation delay depending on the MUX structure.

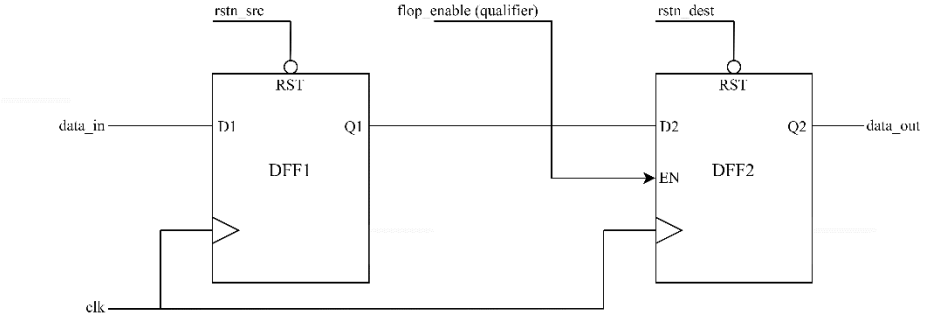


Figure 8 : Flop Enable/Disable mechanism.

Before asserting the reset of the source, the qualifier logic state needs to be flipped such that it selects the constant tied input as the multiplexer output. The qualifier needs to be synchronous with the destination clock domain, so the MUX would not generate any asynchronous output. Inferring these flip-flops with enable pin to register all signals that cross the reset domain could be effective if a single high fan-out qualifier signal can be generated. This can effectively block the entire destination reset domain.

3.1.4 Reset Re-sequencing

In Section 1.2, the general categorization of resets on a chip is discussed. A reset or a set of resets having mutually assertion synchronicity can be grouped under a reset domain and depending on their propagation depth and impact area on the chip, this domain can be labeled as cold PoR, warm PoR, hardware, or software. Additionally, the hierarchy of the scope of impact can be derived from the most extensive to the most localized.

Figure 9 depicts the hierarchy of assertions of reset domains which were illustrated in Figure 2. Here the assertion of a certain domain simultaneously asserts the domain pointed by the arrowhead. For example, assertion of warm PoR domain forces all HW reset domains to assert. This behavior is inherent from the design, and it makes the RDC paths in the direction of the arrow safe by reset assertion order. RDC paths in the opposite direction are vulnerable to cause metastability at the destination.

Doubled headed arrows represents the relationships between reset domains where no inherent assertion order is found. In such scenarios, the design needs to implement and optimize a proper reset assertion sequence such that it minimizes the unsafe RDC crossings count while preserving the functionality.

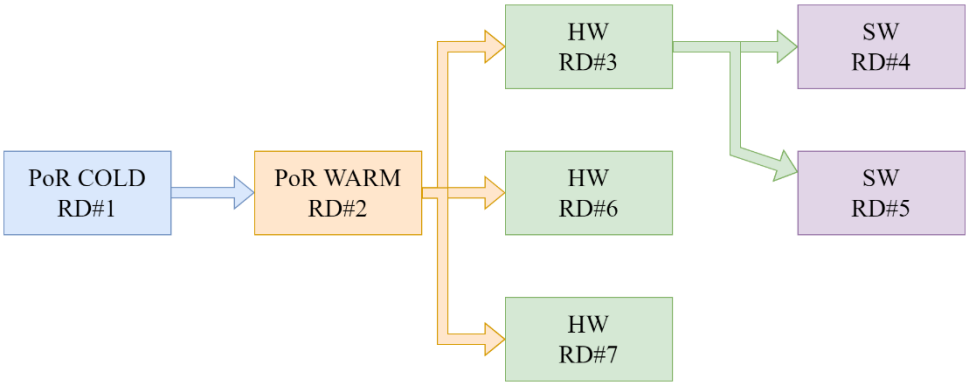


Figure 9 : Initial Reset Assertion Hierarchy

Reset Domain merging can also be performed to minimize the number of asynchronous reset domains if necessary, effectively reducing the number of RDC that needs to be handled.

Figure 10 illustrates a possible reset assertion sequence ordering and reset domain merging fix for the above scenario. Here the HW reset domains RD#6 and RD#7 are merged while implementing a proper reset assertion order from the merged HW reset domains (6+7) to SW reset domain RD#5. By optimizing this approach, a significant number of RDC violations that needs handling can be removed.

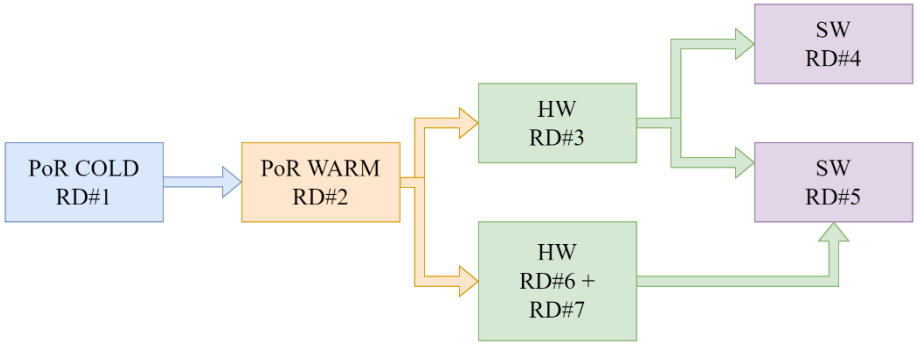


Figure 10 : Altered Reset Assertion Hierarchy

3.1.5 Flip-flop Synchronizer

The most common way to avoid propagation of metastability is to employ one- or multiflip-flop synchronizers across a signal path prone to metastability in clock domain crossings. The implementation of reset domain crossings is illustrated in Figure 11 for the same clock domain. Q1 data are asynchronous with respect to the *clk* upon *rstn_src* assertion which creates a risk of SDFF1 going to metastability. This method allocates a duration of an entire destination clock period (excluding the setup time of the SDFF2) for the output of the first synchronizer flip-flop, Q2, to resolve itself to either legal logic state if metastability occurs.

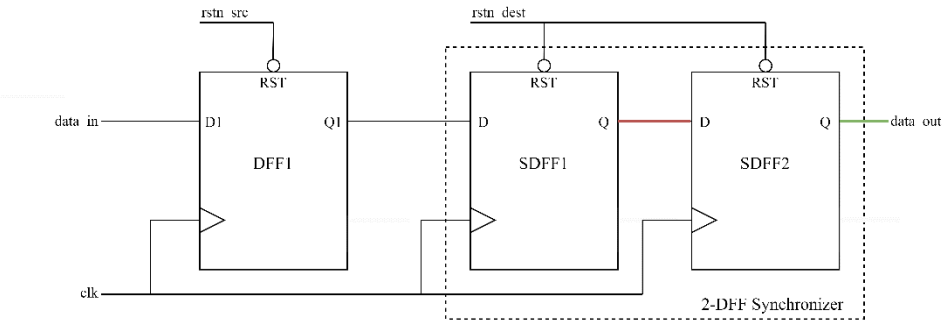


Figure 11 : Flip-flop Synchronizers

Depending on the logic state to which SDFF1 recovers or whether it reaches metastability at all, a random one-cycle delay is introduced for the synchronized value *data_out* as shown in Figure 12. Scenario1 occurs if the first synchronizer output settles to the expected reset logic state within the allowed resolution time, or there is no metastability at all. Scenario2 is possible if the first synchronizer output settles to the opposite of the expected reset logic state. [7]

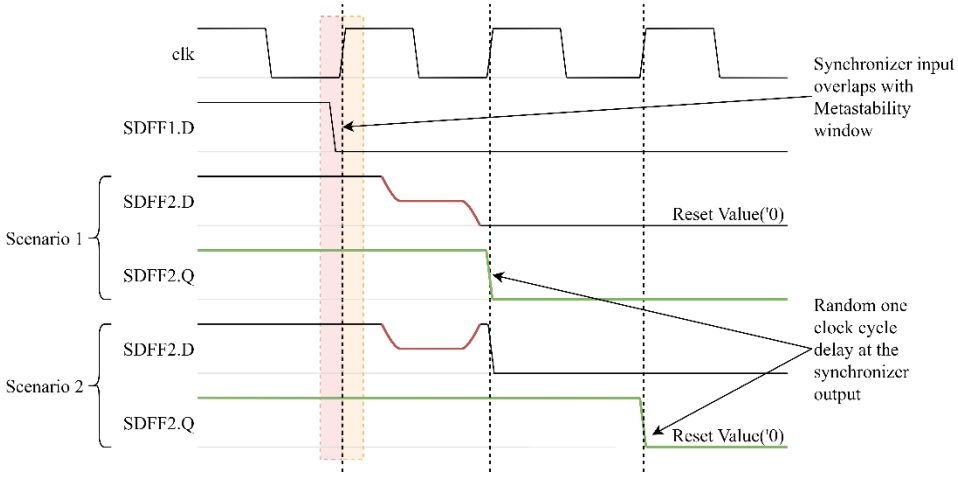


Figure 12 : Occurrence of uncertain one clock cycle delay across a flip-flop synchronizer

VCSG can perform comprehensive analysis of reset domain crossings in the RTL design, identifying all instances where reset signals cross between different domains. In VC SpyGlass RDC, it is required to constraint the enable single in clock-gating, data-gating and flip-flop enable blocking mechanisms, as the qualifier signal for the tool to recognize the safe mechanism properly blocks the asynchronous data input to the destination.

Reset sequence or the assertion order is also required to be defined as a constraint so the tool can skip RDC analysis in safer crossings. VC SpyGlass gives the flexibility for the designer to choose whether the flip-flop synchronizers need to be detected as a solution for RDC.

In summary, various techniques are available to address reset domain crossings in SoCs, each with its own set of advantages and challenges. Designers must carefully consider these factors when choosing an appropriate method for their specific application. In practical scenarios of complex circuits, a combination of multiple methods with a hybrid approach is required to be implemented to achieve an optimum RDC solution. In most cases, it is recommended to optimize the reset assertion order while preserving the expected functionality of the design so that a minimum number of unsafe RDC paths are reported and implement other blocking methods, such as clock or data gating, to resolve the rest of the RDC. The complexity of this approach may exponentially increase with the maturity of the

design. Therefore, considering reset domain crossings early in the design cycle is a recommended practice.

4. Proposed RDC Verification Methodology

In this section, we discuss various methods to identify RDC issues using Synopsys VC-Spyglass tool. In addition, few key constraints are presented to the benefit of VCSG user. Furthermore, constraining of qualifier signals is also presented so that the tool recognizes strategies implemented to counteract RDC issues. Firstly, we will present on block level analysis and then we will move on to hierarchical RDC analysis.

4.1 Block-level RDC Verification

We present a basic generic design which mimics various types of RDC domains of large complex SoC. Then the feasibility of RDC solutions is analyzed for each issue and the suitable method is then implemented. The RDC was reverified after fixing the RTL. In work, we have also enabled glitch and convergence issues due to reset domain crossings, this makes our analysis quite compressive. We would like to share these methods to other users of VCSG so that every design is verified thoroughly.

4.1.1 RDC resolved by reset assertion re-ordering.

There exists an RDC path from rstn1 to rsnt3 and rsnt4 which are 3 resets in different reset domain as shown in Figure 13. Suppose by the implementation of reset generation logic, the resets rstn3 and rstn4 always asserts before the assertion of rstn1. This makes the path safe by the reset assertion order because the destination reset domain undergoes a reset before the source. And this design information is communicated to the tool using the reset order defining constraint, which avoids reporting all RDC violations between the specified source and destination resets.

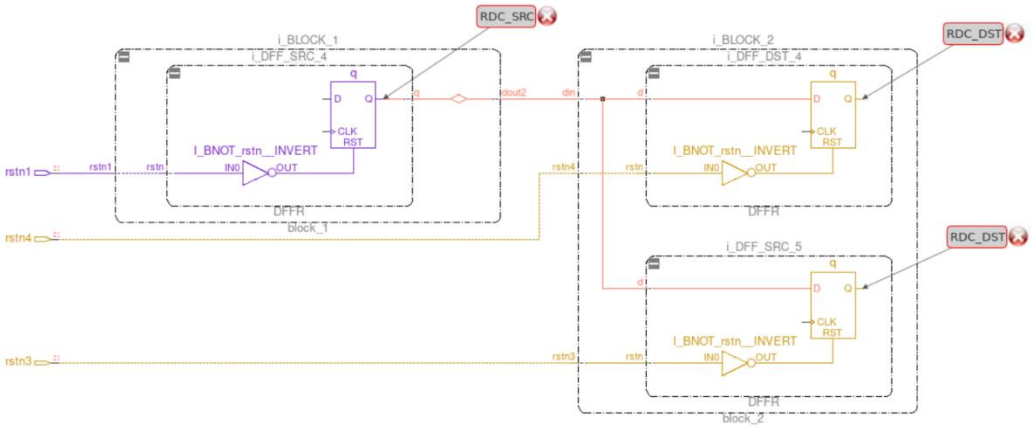


Figure 13 : RDC fixed by reset assertion order.

Reset assertion order constraint for VC SpySlass is as follows where -from_reset is the source reset and -to_reset are the destination resets.

```
set_rdc_define_assertion_sequence -from_reset RSTN_1 -to_reset { RSTN_3 RSTN_4 }
```

4.1.2 RDC with resetless flop chain handled with Clock-Gating

We were advised by Synopsys FAEs to use VCSG instead of SG especially for the analysis of RDC with resetless flops. VCSG can skip N number of resetless flops so that actual issue is showcased to

the designer. Within a sub module, an unsafe RDC path exists from the reset domain of rstn1 to the reset domain of rsnt2. As shown in Figure 14, the tool is configured in such a way that the series of flip flops without a reset along the path is skipped until a resettable destination is encountered. This is done since the outputs of the resetless flip-flop are not observable at the module boundary, and the functionality is unaffected if any of these flip-flops goes to metastability.

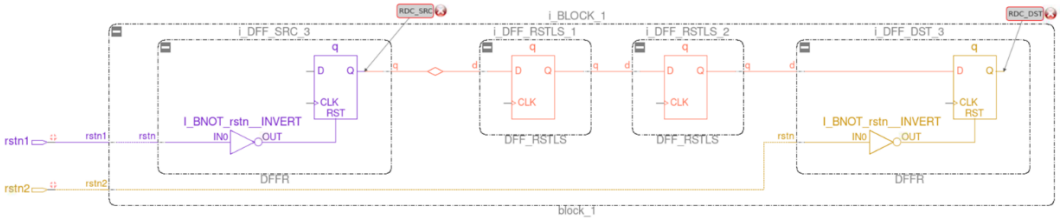


Figure 14 : RDC through resetless flip-flop chain.

This RDC path is blocked by gating the clock of the destination domain, rstn2. As shown in Figure 15, the tool has identified and labeled the clock gating cell, after constraining the clock enable signal as a qualifier.

```
set_rdc_qualifier -object clk_en_top -from_reset RSTN_1 -to_reset RSTN_2 -
depth 2
```

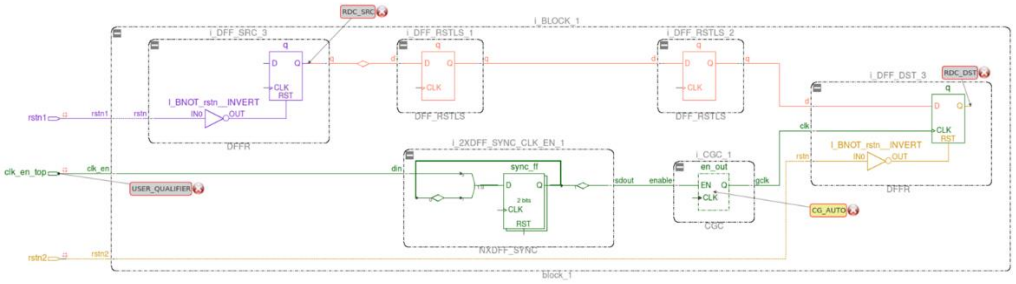


Figure 15 : RDC fixed by Clock-gating.

Inferring a clock-gating cell to the clock network can result this network to propagate metastability.

4.1.3 RDC handled with Data-gating

There is an unsafe RDC path connecting the reset domain of rstn3 to the reset domain of rsnt4. This issue is fixed by gating the data path before reaching the destination domain. As shown in Figure 16, the tool has identified and labeled the AND gating logic which disables data propagation.

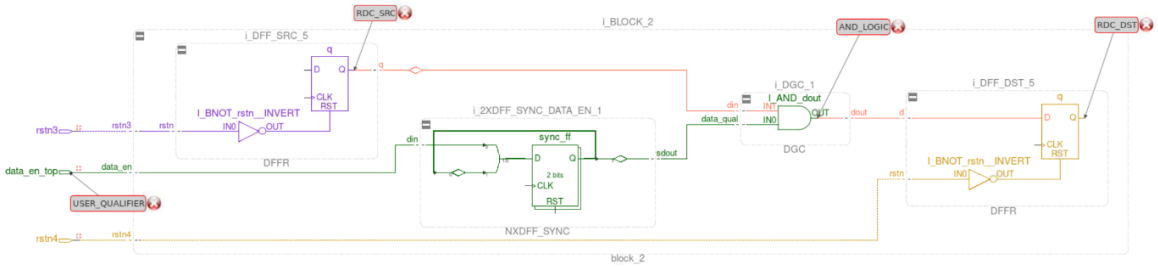


Figure 16 : RDC fixed by Data-gating.

Here the data enable signal needs to be constrained as a qualifier. Usually, the generated enable signals are asynchronous to the destination. Therefore, it needs to be synchronized before used at the gate. This can be done using a flip-flop synchronizer and VC SpyGlass can be configured to consider flip-flop synchronizers as a valid RDC method as follows.

```
configure_rdc_nff_sync -enable
```

And the number of sequential elements from qualifier defined node to the gating cell needs to be specified as the depth as follows.

```
set_rdc_qualifier -object data_en_top -from_reset RSTN_3 -to_reset RSTN_4
-depth 2
```

4.1.4 Clock Corruptions

If the clock enable signal at the clock gate input transitions asynchronously because of the asynchronous reset assertion at its source, it could corrupt the clock network after converging with the clock at the gate. This is caught in the tool as shown in Figure 17.

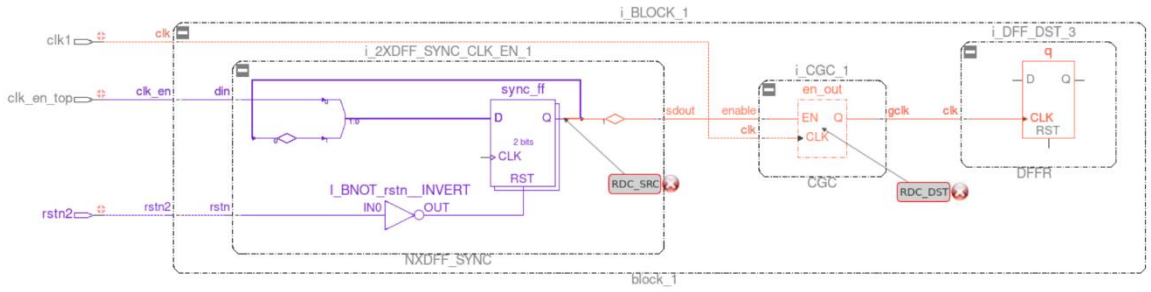


Figure 17 : Clock corruption at CGC due to asynchronous reset assertion.

If the destination clock is inactive before the assertion of source reset, the corruption is not possible, and the clock gating is safe. This behavior needs to be implemented by making necessary design corrections. The impact of clock corruption can be mitigated by using the same destination reset to assert the clock enable synchronizer reset.

4.1.5 RDC Glitch

There are two unsafe RDC paths that combine and propagate through combinational logic as shown in Figure 18. Such paths are prone to glitches since the combined paths are asynchronous to each other.

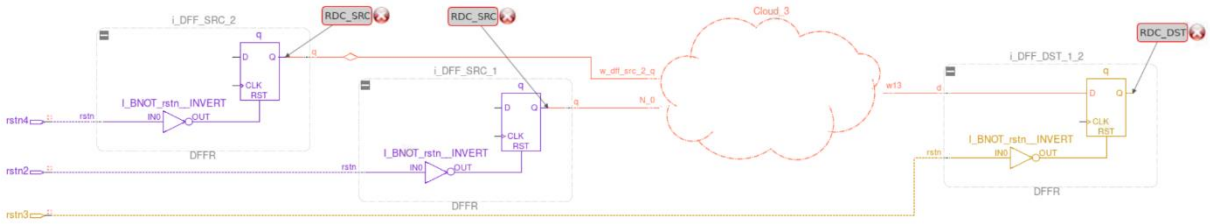


Figure 18 : RDC paths combined before destination.

If a multi-flip-flop synchronizer is used to resolve this unsafe path, even though it seems like the metastability is avoided, the glitch path can increase the MTBF and cause functional failures. This issue is caught in the RDC Glitch analysis stage, as shown in Figure 19.

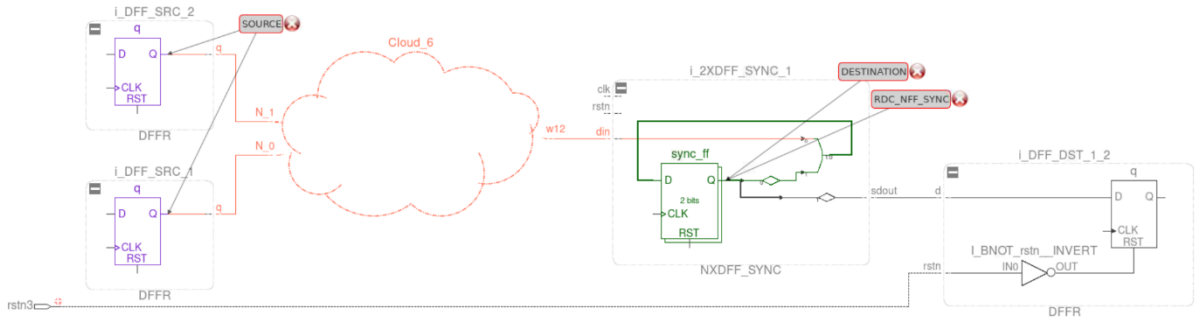


Figure 19 : Combined RDC paths detected as glitch source to destination flip-flop.

Usually, the fix for this involves changing the circuit structure such that this glitch issue does not arise or using clock or data gating mechanism at the destination.

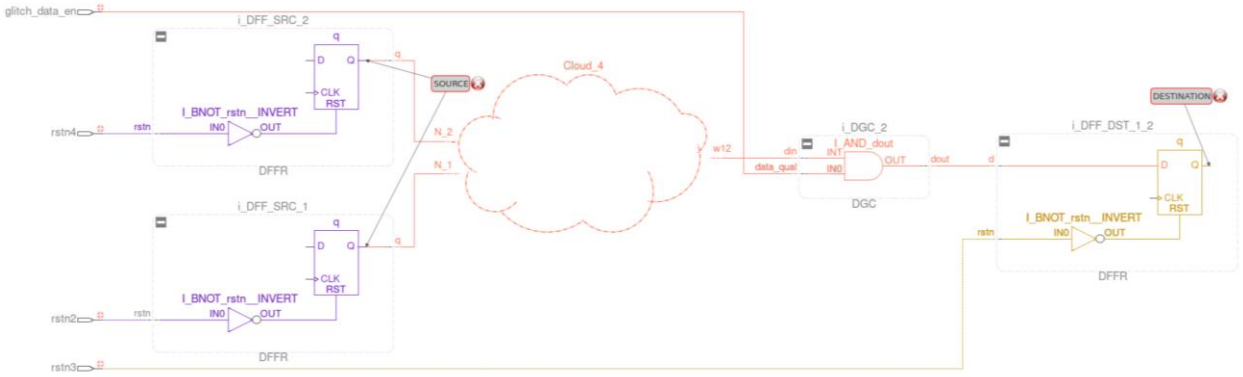


Figure 20 : Glitch potentially blocked by a data gate qualifier.

4.1.6 RDC Convergence

In Figure 20, the unsafe RDC paths from rstn 2 converge and propagate to rstn 3. Here multi flip-flop synchronizers in rstn3 domain are placed before the combinational logic. This resolves the RDC corruption, but now the synchronized paths converge before the destination. It is known that there is an uncertain one-cycle clock delay in flip-flop synchronizers. Therefore, separately synchronized paths may have different delays and can introduce integrity issues at the destination, forcing the circuit to move to an unknown state. Usually, the fix for this involves changing the circuit structure so that this convergence issue does not arise.

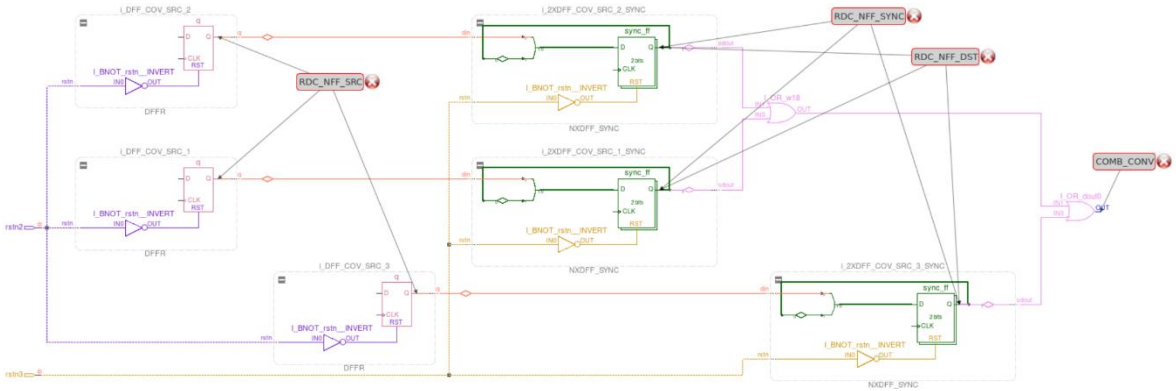


Figure 21 : Separately synchronized RDC paths converge at destination.

In convergence checking, skipping resetless flops is not allowed as flip-flop synchronizers need to be recognized as a method of RDC.

4.2 Top-level RDC Verification

Hierarchical RDC helps manage the complexity of large digital designs by breaking down into manageable units while ensuring that reset signals are properly handled across different levels of hierarchy. This approach enhances design scalability, modularity, and maintainability, ultimately leading to more robust and reliable digital systems. The sign-off abstract models (SAM) of the IP blocks are used in the top environment.

4.2.1 Design under consideration

The Design TOP consists of 3 submodules with complex clock and reset structures, each submodule serving as an excellent candidate to test the Reset Domain Crossings. Data and control signals exist between each module in both directions as shown in Figure 21.

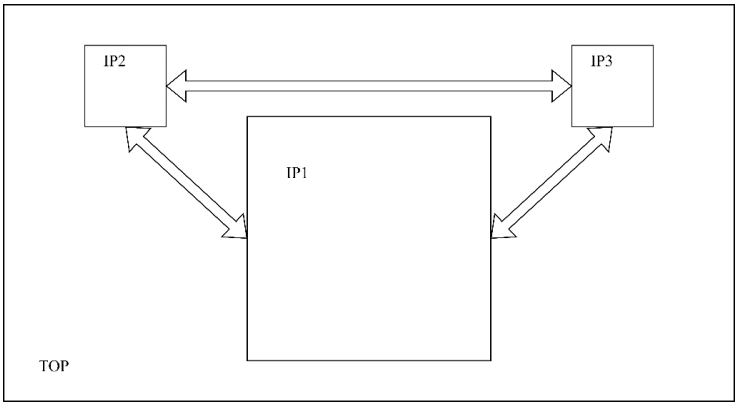


Figure 22 : Top level view of the DUT

Structure and complexity of the design are summarized and tabulated with respect to the clocks and resets of the top and each submodule. The tool is configured such that it detects multi flip-flop synchronizers (two flip-flop synchronizers) automatically as a method of synchronizing RDC paths. Resetless destinations, such as resetless flip-flops, hanging outputs, or output ports, are ignored when checking for corruptions at the block level. The design does not consist of any resetless flip-flops with observable outputs. Therefore, checking such destinations for RDC corruptions is futile. RDC corruptions overlapping with CDC paths are also ignored, since CDC sign-off usually happens prior to RDC and CDC solutions fix RDC corruptions on the same path. Clock corruptions at clock-gating cells due to asynchronous reset assertions are set to be reported.

Table 1: Design Complexity Summary

	Type	**IP1**	**IP2**	**IP3**	**TOP**
Clocks	Total	21	19	8	33
	Primary	5	13	3	14
	Generated	16	5	5	19
Clock Domains		6	4	3	7
Resets	Total	7	7	3	15

	Primary	5	3	3	9
	Generated	2	4	0	6
Reset Domains		2	1	2	4

5. Results

5.1 RDC Analysis and resolution of IP1

IP 1 is the largest submodule inside the TOP. The design summary of IP1 can be extracted from Table 2. It consists of six clock domains and two asynchronous reset domains. For the initial RDC setup check, an SDC file from the synthesis was obtained as the main constraint file. All required clocks and clock groups were constrained, but constraints related to resets were missing.

5.1.1 Solution 1 : Fixing Constraints

The initial step that was performed is applying all design-specific constraints required using SDC/TCL format. Clean RDC verification can be done using a setup with a completely constrained design. Many unnecessary violations will be removed and only valid violations pointing to actual design errors will remain. The following constraints were fixed for IP1,

- Clock constraints to create and specify the primary clock and generated clocks.
- Clock grouping constraints to specify synchronous and asynchronous clock domains.
- Reset constraints to create and specify primary and internally generated resets.
- Reset grouping constraints to specify synchronous and asynchronous reset domains.
- Constraints to set ports/objects to propagate constants in operating mode.

These were fixed using relevant SDC/TCL commands and rerunning the setup check until all the violations related to the setup are solved. No black boxes are found in the design. Otherwise, modeling the boundary of the black box with constraints would be necessary. After the setup check, the RDC corruption check was performed. One of the issues related to the reported results was that RDCs were reported for a particular reset pair, where each reset acts as both the source and the destination. If there is a reset assertion order, then one reset will always be the source reset, and the other the destination reset. This is a design functionality related to the reset generation unit. This is communicated to the tool using constraints to specify set assertion order.

Table 2 shows the summary of the violation counts at each stage. It can be observed that more than 60 percent of the initial violations are resolved after setting the required design constraints.

Table 2: Block - IP1: Violation summary.

Violation	Initial	Constraint fix	Clock-gating fix
RDC Corruptions	G:4, C:79447	G:2 , C:32322	G:2 , C:2558
RDC Potential Corruptions	-	-	G:1 , C:294
RDC Safe by Assertion Order	-	G:2 , C:47125	G:2 , C:47125
RDC Corruptions Blocked	-	-	G:1 , C:29470

5.1.2 Solution 2 : Clock Gating

The remaining RDC violations cannot and should not be prevented with any constraints. They are

actual design bugs that need to be fixed in the RTL. From the RDC methods discussed, the clock-gating method was selected to be implemented within IP1. The reason behind this selection is that the count of RDC paths from one certain reset to another certain reset was significantly high, and data-gating techniques would require many data-gating cells to block all these paths. Multi-flop synchronizers would raise a lot of glitches and convergence issues due to uncertain delays in them. A clock enable-disable signal needs to be generated at the Reset Control unit to disable the clock before asserting the source reset. This signal is constrained as an RDC qualifier so that the tool would identify the paths as safe.

5.2 RDC Analysis of top in hierarchical flows

The aim of this activity was to analyze how the use of hierarchical flow for RDC verification affects the complexity of analysis and parallel verification in bottom-up design flow. The bottom-up RTL design flow involves creating and verifying individual modules or blocks first and then progressively integrating and validating them to form the complete system-level(top-level) design. Hierarchical verification in a bottom-up RTL design flow is efficient because it allows for modular testing of individual components or blocks, which simplifies debugging, and accelerates the identification and fixing of errors before they propagate and complicate at the top-level. Below table summarizes the results of both flattened verification and hierarchical verification.

Table 3: TOP: Violation summary

Violation	**FLAT**		**HIER – Sign-off Abstract Model flow**	
	Initial	**Assertion Order Fix**	**Initial**	**Assertion Order Fix**
RDC Corruptions	G:22 , C:127770	G:11 , C:72799	G:16 , C:48331	G:4, C:34621
RDC Clock Corruptions	G:24 , C:5392	G:17 , C:526	G:3 , C:16	G:2 , C:12
RDC Potential Corruptions	G:7 , C:6581	G:4 , C:4189	-	-

Parallelization of top-level and block-level verification involves verifying the top-level system concurrently with its individual submodules. In the flat verification of the top design, all logic within each IP is synthesized and structural problems are checked. In this case, if the block-level verification sign-off is not given, the unresolved issues inside the block are visible in the top-level analysis reports, which is noisy and adds overhead to the top-level verification. As a solution for this inconvenience, abstract models of submodules, which are simpler and less detailed representations consisting of port interfaces and logic required for top-level clock and reset analysis only, are used at the top-level to enable early validation of top-level integration and interactions while the detailed block-level verification is performed simultaneously. Table 3 shows the reduction in RDC violations reported at the top level after using the hierarchical abstract model approach. Unsafe RDC paths within individual submodules are not reported. And only the RDC paths connecting each submodule are flagged for verification. This reduces the noise and verification overhead at the top-level.

6. Conclusions

In conclusion, the increasing complexity of System-on-Chips (SoCs) requires advanced strategies for managing Reset Domain Crossings (RDC). To tackle these challenges, we have outlined a series of methodologies and VC-Spyglass tool that not only reduce the risks related to RDC but also streamline the verification process at both the block-level and the top-level. The strategic implementation of

blocking techniques, such as clock and data gating, along with proper utilization of VC SpyGlass, has played a crucial role in controlling the spread of metastability and ensuring the functionality of the SoC. This study has emphasized the significance of adopting a hierarchical verification approach and has promoted the early consideration of RDC during the design phase, which contributes to the creation of more resilient and dependable SoCs. In addition, we have enabled glitch and convergence checks for the proper signing off our chips. Our initiatives have led to a substantial decrease in RDC violations and adds a more efficient phase to the current design and verification flow within the company.

7. References

[1] Arora M. (2011) The Art of Hardware Architecture: Design Methods and Techniques for Digital Circuits, Springer New York, chapter 1-2, pp. 1–50.

[2] Ginosar R. (2011) Metastability and synchronizers: A tutorial. In: IEEE Design Test of Computers, pp. 1–13.

[3] Kwok C., Viswanathan P.&Yeung P. (2015) Addressing the challenges of reset verification in soc designs. In: 2015 DVCon US.

[4] Viswanathan P., Takara K., Kwok C. & Ahmed I. (2018) A specification-driven methodology for the design and verification of reset domain crossing logic. In: 2018 DVCon US.

[5] Ahmed I., Nouh K. & Abbas A. (2017) Multiple reset domains verification using assertion-based verification. In: 2017 IEEE VLSI-SoC UAE.

[6] Fawzy M., Elgohary A. & Ibrahim H. (2020) Noise reduction in reset domain crossings verification using formal verification. In: 2020 IEEE East-West Design Test Symposium (EWDTS).

[7] Cummings C.E., Mills D. & Golson S. (2003) Asynchronous synchronous reset design techniques - part deux. In: 2003 SNUG Boston.