RESET VERIFICATION IN SoC DESIGNS

CHRIS KWOK, PRIYA VISWANATHAN AND KURT TAKARA
MENTOR A SIEMENS BUSINESS

Mentor®
A Siemens Business

WHITEPAPER

FUNCTIONAL VERIFICATION

www.mentor.com

INTRODUCTION

Today's SoC designs integrate many design IP blocks from various providers, each with their own implementation of reset. The reset architecture of a digital design can also be very complex. Designs have multiple sources of reset, such as power-on reset, hardware resets, debug resets, software resets, and watchdog timer reset. Errors in reset design can lead to metastability, glitches or other functional failures of the system. Furthermore, complex interactions can occur with the convergence of multiple resets, multiple clocks and multiple power domains. In many cases, this leads to a reset tree that is larger and more complex than the clock tree. Many of the concerns related to clock tree synthesis and load balancing now apply to the reset tree as well. Clearly, it's a challenge to ensure that all sources of reset propagate safely to the intended destinations under all conditions!

Traditionally, simulation has been the primary method used to verify reset behavior, often with a heavy reliance on gate level simulation. However, RTL-level simulation testing is often incomplete, and gate level simulation can only be run late in the design cycle. Even worse, typically reset-related bugs are of a very serious nature, rendering the chip completely unusable. For example, a reset bug may prevent the reset of the design to a known good starting state, making its operation completely unreliable. In more extreme cases, the design may consume too much power during assertion of reset, causing the device to overheat and be permanently damaged. All of these factors negatively combine to cause costly, late-stage design changes; and, in the worst case, multi-million dollar silicon re-spins and time-to-market delays.

COMMON PROBLEMS

We first highlight some of the common reset architecture issues that we have seen in our experience on real world designs. We will separate it into two main categories: (a) issues related to correctness of reset trees and (b) issues related to the usage of resets.

A. Reset Distribution Tree

The first set of reset design issues are related to the incorrect implementation of resets. These problems are usually detected in the reset tree. Figure 1 shows an example—if this reset tree is incorrect, then all the other checking based on the reset sourced in the design will be incorrect, and the chip will not function properly. There is a long list of potential problems that we have seen, and we highlight two common ones here.

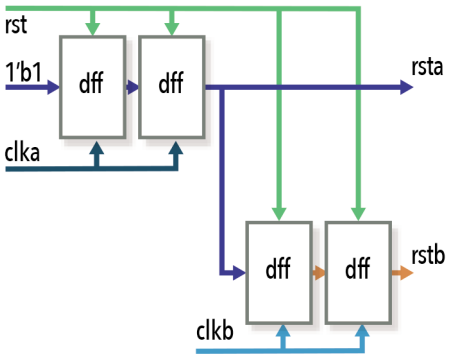

Figure 1: Typical Reset Tree Topology

In general, resets can only be defined as asynchronous or synchronous to a clock. Sometimes, it's being used as synchronous reset for clock1, and asynchronous for clock2. This usually indicates a misunderstanding of the reset in the system. Figure 2a shows a simple schematic in which a reset is used as both asynchronous and synchronous.

Sometimes while the design is evolving, wrong logic could be inserted into the reset tree. Incorrect logic can come in different forms. Some common problems include the addition of tristate buffers and gates such as XOR gate, as shown in Figure 2b and 2c.

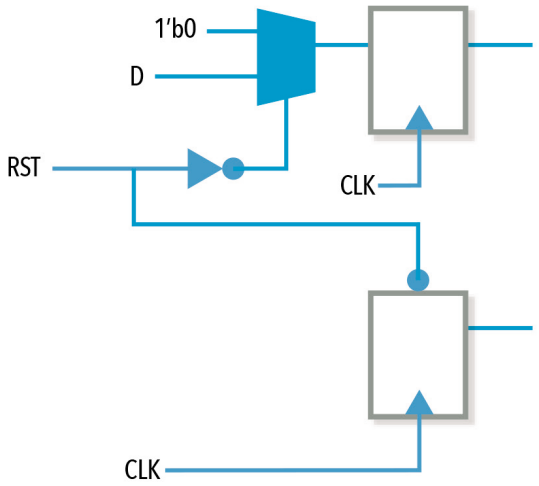

Figure 2a: Reset with Mixed Types

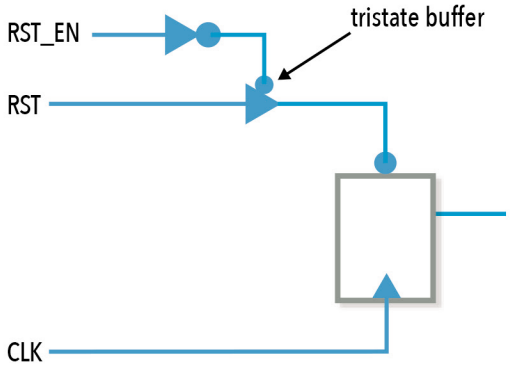

Figure 2b: Unexpected Tri-state

The implementation of the reset distribution network includes the reset sources, reset design hierarchy and the logic elements contributing to the reset logic all of which is detailed in the reset tree generated by tools. Scrutinizing the details of the reset tree information can resolve many such reset design issues as above.

B. Reset Usage

The second set of reset problems is related to usage of the reset signaling to initialize the design.

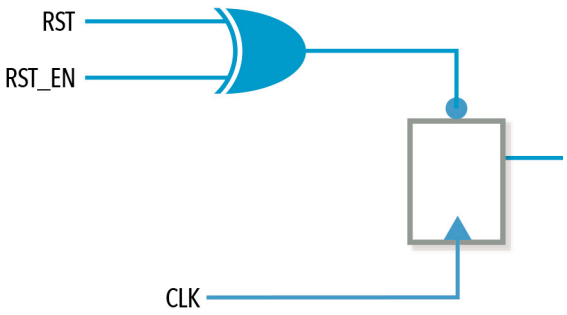

Figure 2c: Unexpected Gate

To describe this set of problems, we need to first define the terms "reset domain" and "clock-domain". A reset domain is characterized by the following attributes – a) Type – synchronous or asynchronous b) Polarity – active low or active high c) Value – set for 1 and reset for 0 and d) the top resetting signal. Multiple synchronous reset domains can be grouped together. Similarly, a clock-domain indicates the clock source of a given register and optionally, the clock polarity. Multiple synchronous clock sources can be grouped together in a single clock-domain.

In a design with multiple reset and clock-domains, it is important to verify that resets are used properly in the context of the clocking scheme design. Specifically, the reset tree annotates the sequential elements in the design with the reset domain information. The clock tree is generated and the clock-domain information is annotated on every register. With every register in the design having a clock-domain and reset domain, the usage of the resets in a particular clock-domain can be determined. With this information, the reset analysis can identify asynchronous reset crossings between sequential elements in the same clock-domain, as well as identify crossings between asynchronous and synchronous resets between registers in the same clock-domain. These paths between resets are called Reset Domain Crossing (RDC) paths.

All asynchronous reset signals should be synchronized to the target clock-domains before being used. If the reset is de-asserted (released) asynchronously, it may violate the reset recovery

time leading to metastability. When an asynchronous reset violates the setup and hold requirements for a register, the metastability due to reset will result in a random delayed release of reset on the metastable registers. Figure 3 shows the waveform for the asynchronous reset problem.

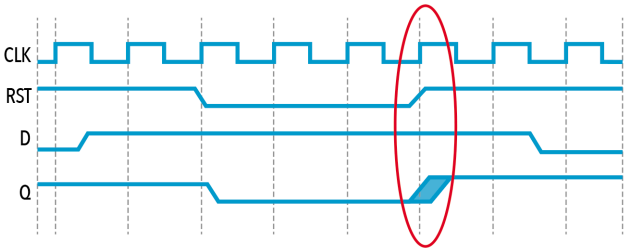

Figure 3

To protect the circuit from metastability due to reset, reset synchronizers needs to be inserted. Figure 4a and Figure 4b show 2 typical reset synchronizers. The external reset signal, RST_n, is active low. When it is asserted, the output of the reset synchronizer will be '0' immediately. When RST_n is released, the output of the reset synchronizer will be '1' in the next or the cycle after. The release of the reset signal is synchronized to the clock, CLK. The synchronizer prevents timing violation when the reset is de-asserted. Figure 4c shows the waveform at the output when synchronizer is inserted. Note that there is no metastability, and the deassertion at Q is delayed by one cycle.

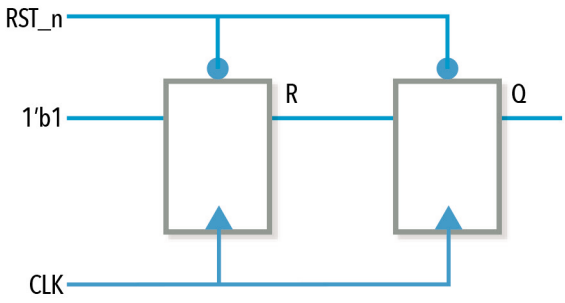

Figure 4a

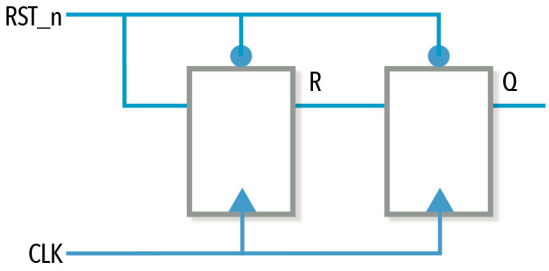

Figure 4b

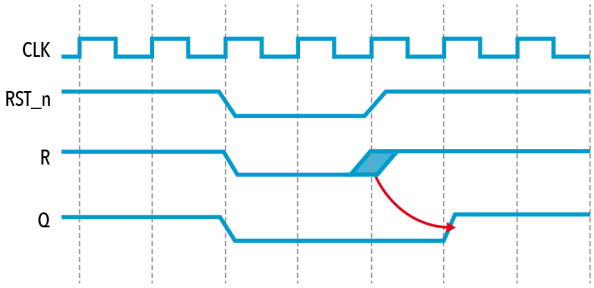

Figure 4c

Once resets are synchronized, designers must ensure that the reset signals are used properly in downstream logic. Two common problems are that resets are used with wrong polarity or clock. Figure 5a shows that the downstream register using the synchronized reset is clocked in a different clock (CLK2) from reset synchronizer (CLK). Clock-Domain Crossing (CDC) analysis will detect a CDC crossing and the problem will be identified during CDC verification. Figure 5b shows that the register reset by the synchronizer is using active high reset, while the synchronizer is an active-low reset. Since this is not a CDC crossing, this error will not be caught by CDC tools, and cannot be caught easily.

C. Reset Domain Crossing (RDC)

The third set of reset problems is related to reset domain crossing (RDC) paths. Most modern chips employ multiple clock-domains, and metastability in such designs due to these asynchronous clock domains "crossing" is a known issue. Advanced tools are available to catch such structural and functional CDC issues. However, even if the data

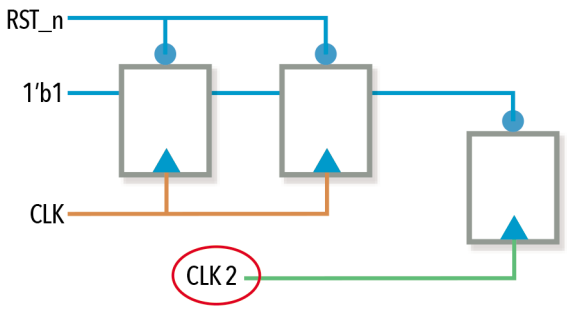

Figure 5a

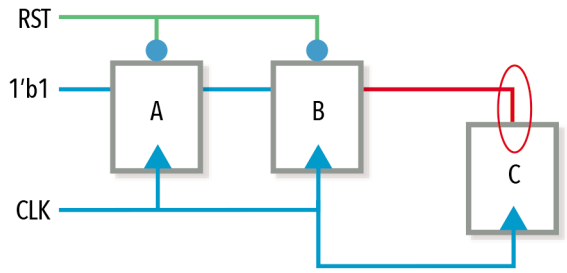

Figure 5b

path is in the same clock-domain, if the reset of the source flop is different from that of the destination register, this asynchronous crossing path can lead to metastability at the destination register. This crossing involves understanding the use of reset domains and their interaction with clock-domains.

The simplest case of RDC is when transmit and receive flops were reset by signals belonging to different reset domains. Figure 6a shows a simple RDC path. If RST1 is asserted while RST2 is not asserted, DFF2 can sample asynchronous data.

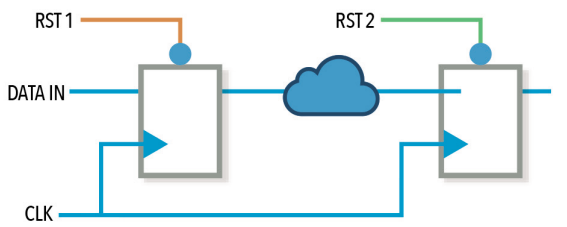

Figure 6a

If the data transition is within the setup and hold window, then the DFF2 output will become metastable, as shown in Figure 6b.

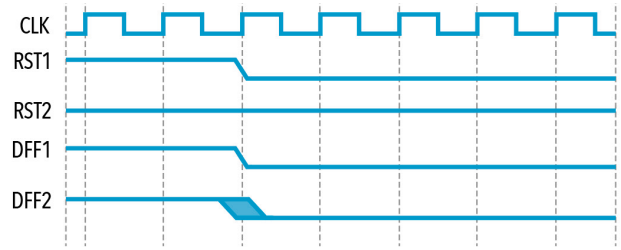

Figure 6b

Design structures used in reset architecture can mitigate RDC problems. One of the RDC synchronization methods is to isolate the input of the receiving domain from the output of the source domain. This requires enable signal to be generated in receiving clock and reset domain which isolates the input of the receiving register when the asynchronous reset of the transmitting domain is asserted. Isolation by gating can happen through the data path or the clock path of the receiving domain, as shown in Figure 7.

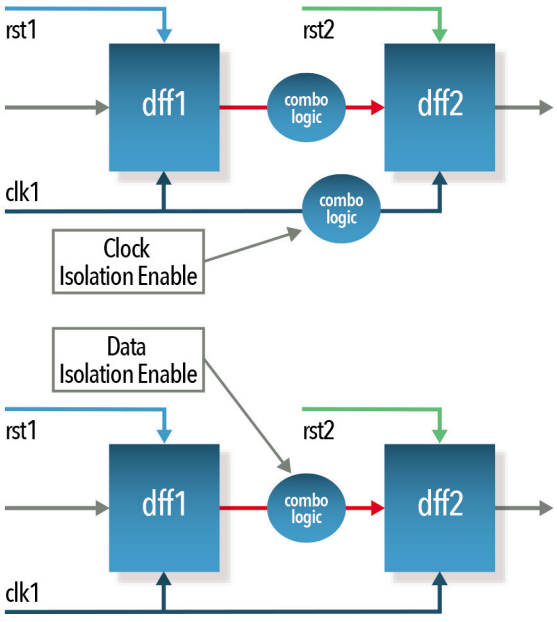

Figure 7: RDC Isolation Methods

In general, chips partition resets through the use of delays in phases to avoid power surges which could burn the chip. This could lead to another problem. Figure 8 shows a simple example. The assertion of RST will cause DFF2 to reset while DFF2 is still in functional state. The inconsistent reset delay between adjacent registers causes incorrect data at the downstream logic. The reset structure assumes that RST is asserted for more than 3 cycles. If RST is only asserted 2 cycles, then when RST deasserts, DFF2 will be corrupted by DFF1 before DFF1 gets the reset signal.

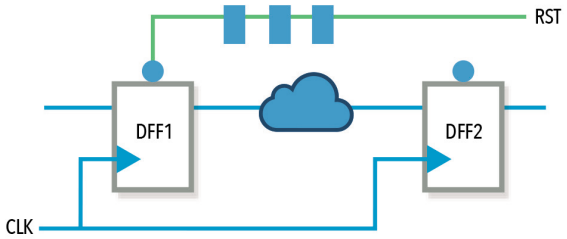

Figure 8

VERIFICATION RESULTS

Because the metastability induced by the reset circuitry issues described above cannot be modeled by simulation, the Questa® Reset Check app – based on CDC analysis technology – can be employed to automatically perform an exhaustive, bottom-up reset tree analysis, then automatically generates and proves assertions that cover numerous reset-specific structural checks. Using real-world DUTs from customers in a variety of industries, Questa® Reset Check was run on 3 designs of varying complexity including a prototype block, a functional controller and a networking design unit (Table 1 above right). Each of the designs show different interactions between the generated clock-domains and the inferred reset domains leading to domain crossing issues. For example, Design 2 has a large number of reset domains, leading to a large number of reset domain crossings. In general, users can customize the analysis by assigning explicit values to the control signals, or by grouping the clock and reset signals explicitly with directives.

Design complexity	*Design 1*	*Design 2*	*Design 3*
Number of registers	305	47016	43622
Number of latches	0	592	0
Number of RAMs	2	0	64
Number of Gate-level modules	0	88	20

Reset/Clock Domains Information	*Design 1*	*Design 2*	*Design 3*
Total number of reset domains	3	36	48
Total number of clock domains	9	5	12
Number of clocks crossing reset domains	2	3	6
Number of resets crossing clock domains	2	5	5

Number of (RDC) Reset Domain Crossings	*Design 1*	*Design 2*	*Design 3*
Data path across same clock domain	3	25766	225
Data path across different clock domains	0	2618	42

Results of Static Analysis	*Design 1*	*Design 2*	*Design 3*
Missing asynchronous reset synchronizer		X	X
Unexpected gate in reset tree	X	X	
Reset signal used as asynchronous and synchronous			X
Good asynchronous reset synchronizer		X	

Table 1: Summary of Reset Verification

Other static reset issues found in the designs included:

- Unused user specified resets
- Conflicts between specification and reset usage
- Combinational logic before asynchronous reset synchronizer
- Asynchronous reset used incorrectly in data

The reset domain crossing across data path registers in the same clock-domain is a RDC violation. Figure 9a shows a simple RDC violation with start register 'mask' reset asynchronously by a signal 'rst' driving the end register 'rx_masked_data' reset asynchronously by a different signal 'clr'. Both the registers are clocked by the output of a scan mode multiplexer thus belonging to the same clock-domain.

Figure 9b illustrates a reset used synchronously and asynchronously in a design. The highlighted path shows the top primary reset input rst_HT_L resetting the first register asynchronously and resetting the second register on the right hand side through a synchronous reset mux select.

CONCLUSION

In this article we presented several common reset problems, the corresponding reset verification challenges, and proposed a comprehensive solution to address these challenges. We have also demonstrated the solution's effectiveness through verification of several customer designs, and presented the issues discovered. We also showed that many of these issues are hard to catch bugs that cannot be caught easily with simulation or traditional verification techniques.

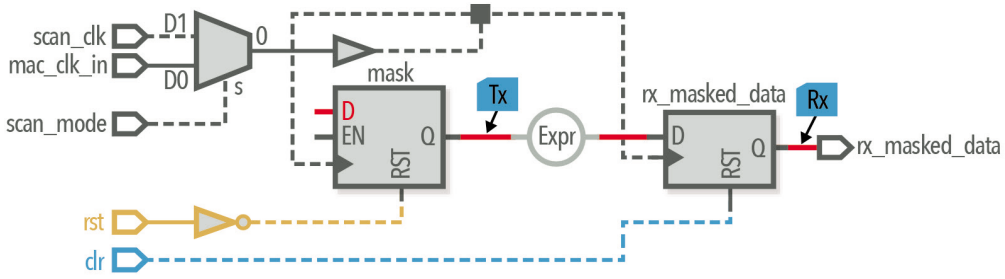

Figure 9a

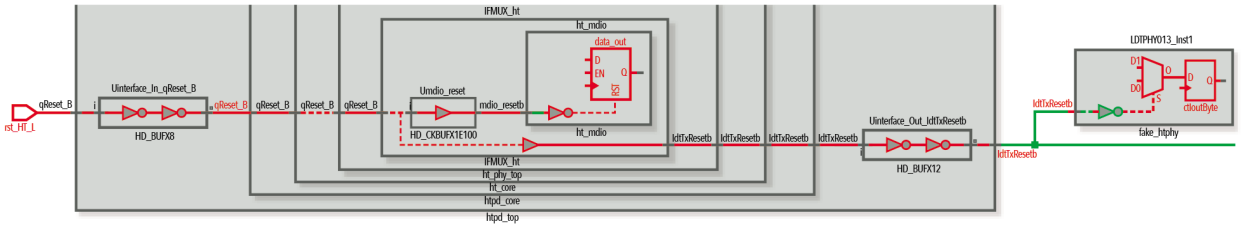

Figure 9b

For the latest product information, call us or visit: www.mentor.com

Corporate Headquarters
Mentor Graphics Corporation
8005 SW Boeckman Road
Wilsonville, OR 97070-7777
Phone: 503.685.7000
Fax: 503.685.1204

Sales and Product Information
Phone: 800.547.3000
sales_info@mentor.com

Silicon Valley
Mentor Graphics Corporation
46871 Bayside Parkway
Fremont, CA 94538 USA
Phone: 510.354.7400
Fax: 510.354.7467

North American Support Center
Phone: 800.547.4303

Europe
Mentor Graphics
Deutschland GmbH
Arnulfstrasse 201
80634 Munich
Germany
Phone: +49.89.57096.0
Fax: +49.89.57096.400

Pacific Rim
Mentor Graphics (Taiwan)
11F, No. 120, Section 2,
Gongdao 5th Road
HsinChu City 300,
Taiwan, ROC
Phone: 886.3.513.1000
Fax: 886.3.573.4734

Japan
Mentor Graphics Japan Co., Ltd.
Gotenyama Trust Tower
7-35, Kita-Shinagawa 4-chome
Shinagawa-Ku, Tokyo 140-0001
Japan
Phone: +81.3.5488.3033
Fax: +81.3.5488.3004

Mentor® A Siemens Business

MSB 06-18 TECH17460-w