Avoiding FPGA Metastability with Reset Domain Crossing Analysis

Abstract

Reset Domain Crossing (RDC) analysis is becoming just as important as CDC analysis. Designers need to be made aware of those points in a design where data is actively received from a source that could be affected by a different reset sequence, or a reset signal based in a different clock domain. This paper will provide tips and tricks to improve reliability on of your FPGA designs.

When resets were global and just used at startup, there was little concern about resets causing metastability, just as there was no concern about Clock Domain Crossings (CDCs) when there was only one clock. But things today are more complex. SoCs include distinct regions that can operate independently and go dormant or even power down when not in use, even if they use the same clock as other regions. Other areas may be reset as part of normal system function. Bringing a dormant region back to life can mean introducing unusual or unpredictable timing issues that can cause metastability in the region that didn't go dormant. Worse, this kind of issue is intermittent and difficult to diagnose, even more so than the more familiar CDC issue. Many current static timing analysis (STA) methodologies don't make use of timing arcs for reset-to-q, even when that would help identify reset paths that don't meet timing. That's why Reset Domain Crossing (RDC) analysis is becoming just as important as CDC analysis. Designers need to be made aware of those points in a design where data is actively received from a source that could be affected by a different reset sequence, or a reset signal based in a different clock domain.

The Blue Pearl Software approach to Reset Domain Crossings (RDC) is an extension of our work on Clock Domain Crossings (CDC) and is covered by the same license. RDC addresses paths that pass through the asynchronous set and reset pins of registers. As with CDC, our primary concern is either avoiding metastability in the circuit, or making sure that any metastability that does arise is properly synchronized before being used elsewhere in the circuit. We focus on identifying paths that cannot be subsequently checked with Static Timing Analysis (STA) because they involve interactions between different domains. The analysis takes place on RTL designs early in the design cycle prior to detailed synthesis and simulation when specific timing information is not yet available.

Metastability can occur either on reset assertion or deassertion. Reset assertion metastability can occur when a source register is asynchronously set or reset, and the resulting value arrives at the data pin of a destination register within the setup and hold window of that register's clock (see Figure 1). In effect, the asynchronous path through the first register is part of a combinational path to the data pin of the destination.

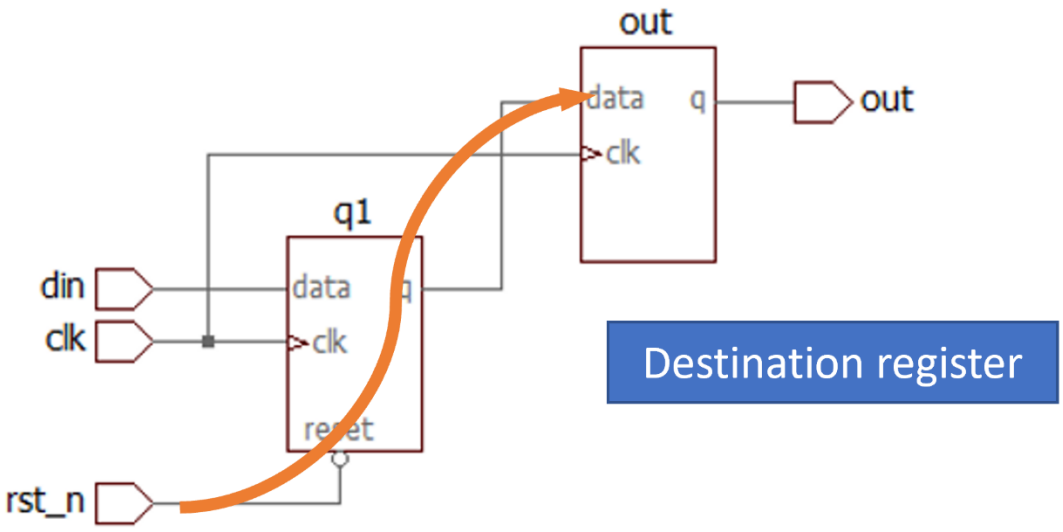


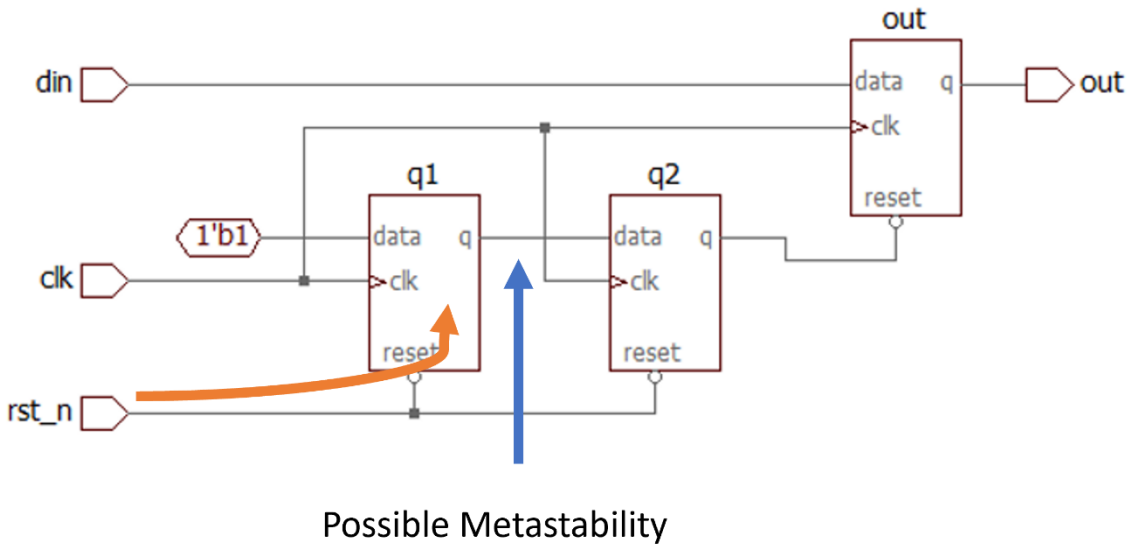
Figure 1: Reset assertion analysis

Note that the clock of the source register (*q1*) is irrelevant for reset assertion analysis. What matters is the clock of the destination register (*out*), and whether the reset signal *rst_n* originates from (or has been already synchronized into) the same clock domain as the destination clock. Also, a destination register that is itself asynchronously set or reset by the assertion of the same reset signal will not give rise to metastability, as the asynchronous set/reset will override the clocking in of the data pin.

While reset assertion metastability can be important in some designs, particularly if some asynchronous set or reset pins are used for system purposes, it may not matter in designs where asynchronous reset pins are only used for a global reset. A global reset will often be asserted over multiple clock cycles, giving any metastability time to die out and the circuit time to settle into a reset condition. More significant is the reset deassertion behavior when the circuit comes out of a reset state. It is important to have the circuit come out of reset in a predictable state.

Reset deassertion metastability can occur when an asynchronous reset is removed during the recovery time window of a register that has its data input at 1 (or an asynchronous set removed from a register with its data input at 0). If the reset (or set) is removed too close to the edge of the clock, the register may not have time to complete the transition to the new value. Unlike reset assertion metastability, the metastability here occurs at the output of the register being reset (see Figure 2).

Figure 2: Reset deassertion metastability



Note that neither the *q2* nor *out* in Figure 2 are subject to reset deassertion metastability. The data input to *q2* is held at 0 while *rst_n* is asserted, so there is no transition on the output of *q2* when the reset is deasserted. The reset signal at *out* is deasserted synchronously to the clock *clk* in the path from *q2*, so it should be outside the recovery time window and can be checked as a synchronous path in static timing analysis.

While the output of *q1* in Figure 2 can be a source of deassertion metastability, the register *q2* acts as the synchronizing register of a double-register synchronizer, so that any metastability is properly synchronized and will not propagate to the rest of the circuit.

Some designs may explicitly synchronize resets into the clock domain where they will be used. Figure 3 shows an example where an external reset (shown in red) is synchronized into two different clock domains corresponding to clocks of two different frequencies. In this example, the reset may be asserted asynchronously in each domain, but it is explicitly deasserted synchronously to the destination clock. On deassertion, a double register in the *rst_gen* block is used to synchronize the reset deassertion to the destination clock, thus ensuring that the system registers will not experience metastability on deassertion.

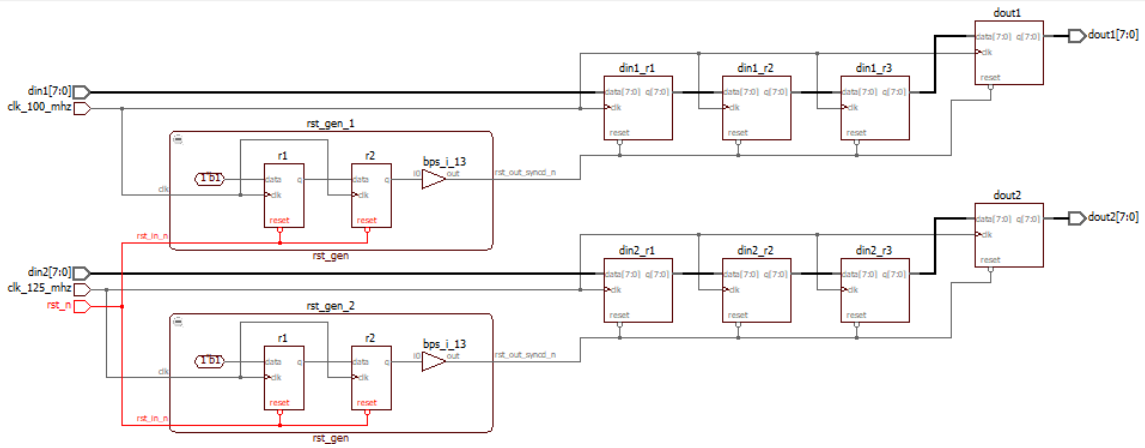


Figure 3: Reset deassertion synchronized into two different clock domains

Synchronous resets

Synchronous resets are frequently implemented as multiplexors in the cone of logic leading to the data input of a register, meaning they are covered in the standard CDC analysis. In general, they should be synchronized into the same clock domain as the receiving register to avoid metastability. Figure 4 shows an example of a synchronous preset and clear signal, and the possible metastability if these signals arrive asynchronously to the destination clock.

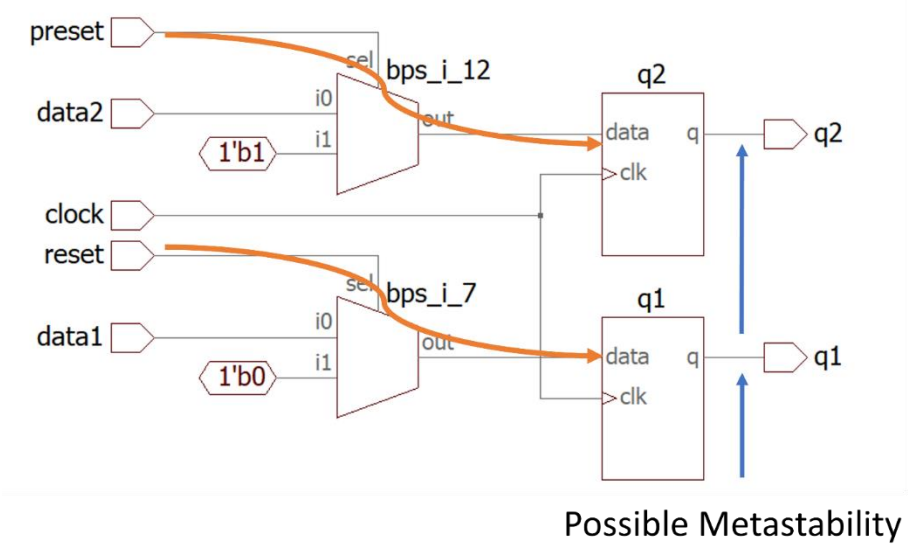


Figure 4: Synchronous sets and resets and potential metastability

The Visual Verification Suite Approach

Treating RDC analysis as part of CDC analysis with extended checks should ensure that either the reset signal is properly synchronized into each clock domain where it is used, or that potential metastabilities are flagged for further analysis. Note that resets that are properly synchronized into the domain where they are used will not produce a CDC violation, but such paths that pass through asynchronous set or reset pins on DFFs will still need to be checked with a static timing analysis tool or similar solutions to make sure they meet timing before design signoff.

Future directions

Blue Pearl Software is participating in the Accellera CDC Working Group that is working to standardize both Clock Domain Crossing and Reset Domain Crossing interfaces. We expect to support this emerging standard in the future.

About Blue Pearl Software

Blue Pearl Software, Inc. is a leading provider of DO-254 compatible design automation software for ASIC, FPGA and IP RTL verification. Our customers are RTL managers and developers in military, aerospace, semiconductor, medical, communications and safety critical design companies. The Visual Verification™ Suite speeds block and project level verification with advanced integrated RTL structural and formal linting, constraint generation and clock domain crossing analysis. Our usability is unmatched in the industry and can help your design team accelerate development and produce high reliability designs. To learn more about Blue Pearl visit www.bluepearlsoftware.com