

Solving FPGA Clock-Domain Crossing Problems: A Real-World Success Story

Timothy Paige, North Pole Engineering, Inc. (tim.paige@honeywell.com) Gordon Braun, Honeywell International Inc. (gordon.c.braun@honeywell.com) Chris Rockwood, Mentor Graphics Corp. (chris_rockwood@mentor.com)

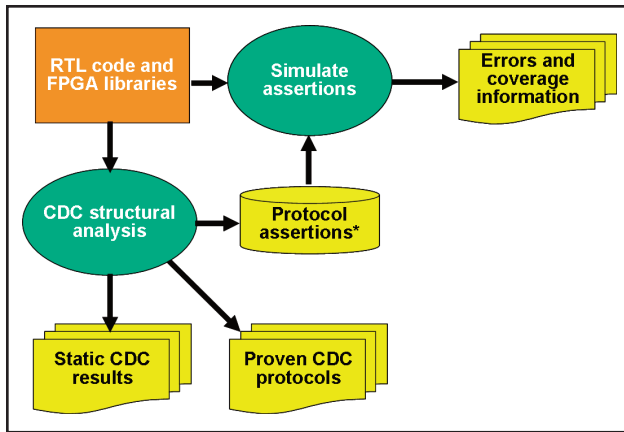


Figure 1. CDC analysis flow

Today's high-capacity FPGAs enable a level of integration that was recently possible only in large ASICs. With many types of processors, complex functions, and interfaces incorporated into FPGA designs, the number of asynchronous clocks has increased rapidly. Most problems associated with clock-domain crossing (CDC) paths cannot be found using traditional simulation and static timing analysis; therefore, a dedicated CDC verification solution is needed. The use of third-party IP further complicates the CDC verification problem. We present here a real-world example of CDC verification on an FPGA design containing third-party IP at Honeywell.

Problem Statement

Although the FPGA design had been verified in simulation using our normal methodology, testing in the lab quickly became frustrating. The device malfunctioned unpredictably, and using the FPGA vendor's debug tool to probe the part seemed to only move the problem around. After two weeks of unsuccessful debugging in the lab, we began to suspect that the problem might be related to CDC signals associated with a specific interface.

CDC Static Analysis

We made the decision to evaluate a CDC analysis tool, initially targeting the IP block connected to the interface in question. After less than one day of work with Mentor's 0-In® CDC tool, we had an initial set of results that identified several problem areas.

Automatic identification of clocks

The tool's ability to automatically identify clocks allowed us to gain a better understanding of the design even during the setup phase. The IP block has three clock inputs, but initially it was not clear whether any relationships between the clocks were required. CDC analysis quickly revealed that two of the three clocks must be synchronous (not necessarily identical, but with a fixed relationship to each other) for the design to function properly. Grouping those two clocks yielded a manageable set of CDC paths to review.

Missing and incorrect synchronization

Our suspicion that there might be problems in the area of the IP block was quickly confirmed, as several instances of missing and incorrect synchronization were reported. The paths reported by the tool as violations fell into three different categories:

1. Obvious design problems.

(a) Missing synchronizers. We were able to convince the IP vendor right away that synchronizers required on some signals were not present and needed to be added (see Figure 2).

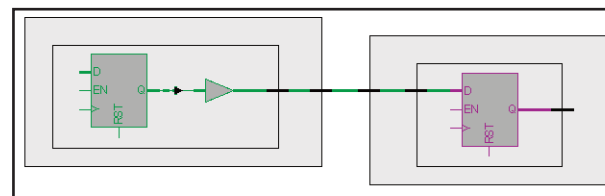


Figure 2. Missing synchronizer (Protocol: transmitting signal must be stable when receiving register is loaded)

(b) Incorrect synchronization. We also had several instances of another common problem: combinational logic in the transmitting domain feeding a synchronizer in the receiving domain (as in Figure 3). A common misconception is that any signal from the transmitting domain is safe to use as the input of a synchronizer on an asynchronous clock, but that is not the case because the output of

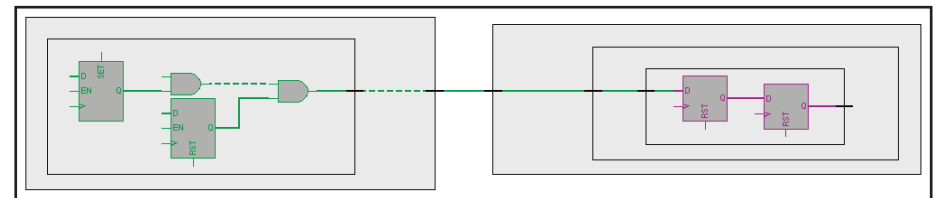


Figure 3. Combinational logic before synchronizer (Protocol: output of combinational logic must never glitch)

combinational logic fed by multiple registers can glitch after each clock edge before settling. Static timing analysis ensures that the output of the combinational logic settles in time to feed registers in the same clock domain, but the first register of a synchronizer on an asynchronous clock can capture these glitches. A proper synchronizer should be fed by only one storage element in the transmitting domain.

2. Transmitting signals that are known to be stable during normal operation. Some signals, such as software configuration registers that are set up once during initialization, are safe to use in other clock domains. It was easy to create waivers for these paths using the tool.

3. Questionable cases. It appeared that some of the transmitting signals should be stable during normal operation, but because we were not the designers of the block it wasn't clear whether these reported violations were safe to waive. CDC protocol checking in simulation (described in the next section) is the best solution for signals whose behavior is not fully understood during the static analysis process.

Static analysis of different operating modes

The IP block has two operating modes. When the design was forced to run in a single mode by setting a configuration register output to a constant value, some unexpected violations were reported. Our investigation of these paths revealed that some logic associated with the disabled mode appeared in the reported CDC paths. This turned out to be a design problem: the IP vendor had failed to disable some paths that should have been used only in the other mode. The visualization capabilities of the CDC tool made this problem easy to find.

CDC Protocol Analysis

Our initial success was based only on static analysis of the design, which is merely the first step in a complete CDC verification process. Every clock-domain crossing (even if it includes a proper synchronization structure) has a transfer protocol that must be followed in order to guarantee that signals will always be properly transferred between domains. Our CDC tool provided two ways to verify the protocols. The static analysis process includes a built-in formal engine that proves some of the protocols and a set of protocol assertions for the remaining crossings. When those assertions are run in simulation, they flag any protocol errors and collect coverage information to indicate how well each crossing has been tested.

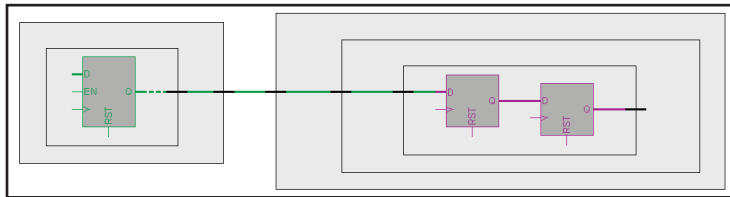


Figure 4. Proper 2-DFF synchronization
(Protocol: minimum pulse-width check on transmitting signal)

Protocol checks for “good” synchronization

For the 2-DFF synchronizer in Figure 4 (and for other similar structures, such as pulse synchronizers), the protocol dictates that the transmitting signal must be held stable for some number of clock cycles in order to guarantee that the signal is

properly transferred into the receiving clock domain. That minimum number of clock cycles depends on the frequencies of the clocks; if the transmitting clock is always slower, the number becomes 1 and the protocol is trivially “proven.”

In our design, most of the simple stability checks were proven either by specifying the clock ratios (for paths from a slower clock to a faster clock) or by the formal engine built into the CDC tool. The remaining protocol checks were verified in simulation.

Protocol checks for paths without synchronizers

Verifying CDC protocols for paths without synchronization is especially important. If the transmitting signal is stable during a window of time around each clock cycle on which the receiving register is loaded, the crossing can be considered safe. For the signals identified during the static analysis that appeared to be stable during normal operation, we simulated the associated protocol assertions. Some of them indicated protocol violations only during initialization, indicating that the waivers created for those paths were appropriate.

The protocol checks for some paths without synchronizers fired during normal operation of the design (as in Figure 5), indicating that design changes were needed to synchronize those signals. The IP block developers initially believed that only a subset of these protocol violations represented design problems and resisted making changes for some of the signals we identified, but we were eventually proven correct because the device continued to malfunction

under certain conditions until the remaining signals with protocol violations were fixed.

Reporting and Visualization

The CDC tool has a GUI that allows clock trees and all reported results to be viewed in schematic form, with links to the associated RTL code. Crossings with the same start points are grouped together, which simplifies the process of reviewing the results. Asynchronous clock domains are color-coded in the generated schematics and in the RTL code; the ability to easily see which clock domain is associated with each signal made the analysis much easier. These capabilities also allowed us to share our results with the developers of the IP block and work with them to analyze the problems.

Summary of Results

We quickly and successfully added a CDC analysis tool to our FPGA verification flow. Within a week, we had reviewed the initial results with the IP vendor and obtained the first set of design changes. The device ran much better after just one week of CDC analysis than it had after two previous weeks of unsuccessful debugging in the lab. It was especially impressive that we were able to obtain actionable results from the CDC tool in less than one day.

Our experience has convinced us that complete CDC verification is essential for today’s FPGA designs. The increasing usage of third-party IP blocks underscores the importance of incorporating CDC analysis into FPGA verification flows.

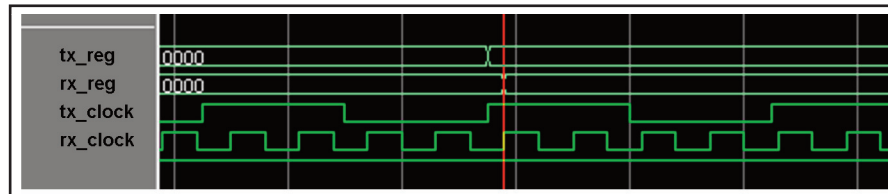


Figure 5. CDC protocol violation for path with missing synchronizer