Choosing the Right Verification Technology for CDC-Clean RTL Signoff

By Pete Hardee, Director, Product Management, System & Verification Group, Cadence

Modern system-on-chip (SoC) designs typically contain multiple asynchronous clock domains. Clock domain crossing (CDC) signals, those which traverse these domains, are often subject to metastability effects that can cause functional errors. Traditional methods like RTL simulation or static timing analysis alone are not sufficient to verify correct data transfer across clock domains. As a result, many CDC-related bugs go undetected until the post-silicon verification stage, necessitating costly re-spins. The Cadence® JasperGold® Clock Domain Crossing App provides a complete solution that automatically infers CDC intent from the design and comprehensively analyzes structural, functional, and reconvergence issues. The JasperGold CDC App also provides an integrated debug environment with advanced debugging options.

Contents

Introduction

CDC verification has been a necessary task in digital semiconductor design ever since chips have been complex enough to need multiple asynchronous clocks, or large enough for perturbations of single propagated clocks to start causing metastability-related problems.

Once the preserve of back-end engineers concerned with chip-level netlist signoff, CDC verification is now a fast-growing concern for verifying IP blocks and subsystems at the RTL stage, for reasons we are about to explain. First, it's worth quickly recapping what the metastability problem is, since some front-end designers and verification engineers may not be as familiar with the phenomenon as perhaps they should be. See Figure 1 for a diagrammatic explanation. When a signal is registered in CLKA clock domain, and then crosses domains to be registered in CLKB domain, if CLKA and CLKB are asynchronous then there is no way to guarantee that the data signal's transitions don't violate the flip-flop's setup and hold time requirements. Changing the input to the flip-flop in CLKB domain too close to the clock transition can result in the flop output going into a metastable state—neither logic 1 or 0—for an indeterminate time.

This metastability effect can of course cause functional error, if not corrected in the circuitry. The usual method to prevent metastability propagating is to insert a synchronizer circuit. The simplest and most common synchronizer type is nDFF—a chain of n flip-flops in the receiving clock domain (a simple n=2 nDFF is shown in blue in Figure 1, but more exotic synchronizers exist such as FIFO based, MUX based, and synchronizers based on custom handshake protocols). Metastability problems can also occur when CLKA and CLKB are not asynchronous by design, but the non-deterministic relationship between sending and receiving clock edges are facets of clock tree implementation—perturbations of the same clock (delays, skew etc.), or between derived clocks, as they propagate across the chip. Sometimes, these problems are not eliminated by design using synchronizers, but can be prevented by careful implementation with strict attention to worst-case and best-case timing constraints between sending and receiving flops. Such fixes are inherently less reliable, are specific only to that chip implementation, and need to be redone each time the design is reused in a different chip.

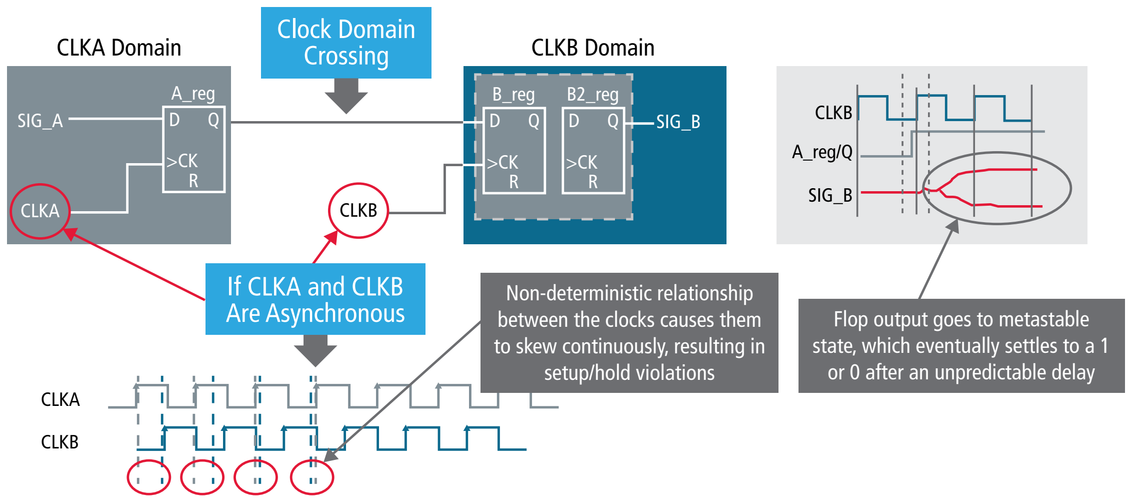

CLKA Domain
Clock Domain Crossing
CLKB Domain
A_reg
SIG_A
D
Q
>CK
R
CLKA
CLKB
B_reg
B2_reg
SIG_B
A_reg/Q
If CLKA and CLKB Are Asynchronous
Non-deterministic relationship between the clocks causes them to skew continuously, resulting in setup/hold violations
Flop output goes to metastable state, which eventually settles to a 1 or 0 after an unpredictable delay

Figure 1: Metastability effect at clock domain crossings

Hence, the need to design robust IP blocks that can be reused for multiple different chip implementations is one of two major reasons why RTL developers and verification engineers are paying much more attention to CDC issues. The second reason is that we're seeing an increase in the number of asynchronous clocks in today's designs, to minimize dynamic power, since each function block is designed to operate at the slowest clock frequency that meets the performance requirement. Designing for low power has significantly increased the number of clock domains on a typical chip, and IP blocks must now be designed for use with a range of different clock frequencies, depending on the power/performance profiles for the target chips. Today's CDC verification problem has moved from chip-level netlist signoff to **ensuring CDC-clean reusable RTL designs.**

Intuitive CDC Verification Steps

The JasperGold CDC App is a comprehensive CDC signoff solution that guides the user through the stages of CDC verification in the following logical and efficient steps:

1. Design and CDC setup
   – Clock setup
   – Signal configuration
   – Automatic clock domain identification
   – Automatic clock domain crossing pair identification
2. Comprehensive structural checks
   – Automatic detection of standard synchronizing schemes including nDFF, MUX, handshake, and FIFO
   – Support for user-defined synchronization schemes
   – Convergence and reconvergence
3. Functional protocol checks
4. Metastability-aware formal and simulation flows
5. Reset domain checks
6. Violation debug, waiver handling, and reporting
   – Automatic waiver flow with waiver validation
   – Efficient report generation

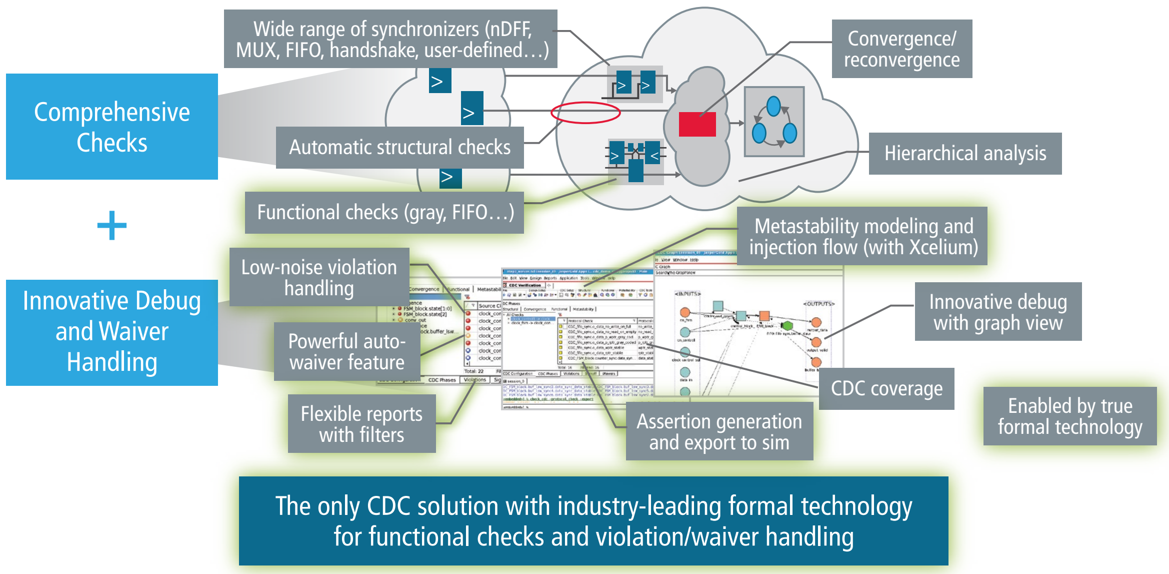

Comprehensive Checks
+
Innovative Debug and Waiver Handling
Wide range of synchronizers (nDFF, MUX, FIFO, handshake, user-defined…)
Automatic structural checks
Functional checks (gray, FIFO…)
Convergence/ reconvergence
Hierarchical analysis
Metastability modeling and injection flow (with Xcelium)
Low-noise violation handling
Powerful auto-waiver feature
Flexible reports with filters
Innovative debug with graph view
CDC coverage
Assertion generation and export to sim
Enabled by true formal technology
The only CDC solution with industry-leading formal technology for functional checks and violation/waiver handling

Figure 2: JasperGold CDC App: Comprehensive checks with formal-powered debug

Design and CDC Setup

Clocks can be defined easily using scripts or using the JasperGold CDC App's intuitive setup GUI. Different clocks are assumed to be asynchronous unless the user explicitly specifies their relation. If clocks have already been defined in the JasperGold platform for use with other formal verification apps, those clock definitions can be reused in the JasperGold CDC App. Clock definitions can also be derived from synthesis design constraints (SDC) files. Note that since RTL CDC analysis is performed exhaustively with any combination and arrival order of relevant clock and signal edges, sub-cycle timing is not relevant. Hence, SDC constructs for modeling perturbations of propagated clocks (e.g., -waveform) are ignored. This is consistent with our goal to verify correct-by-design CDC-clean RTL, not RTL that happens to work for certain timing constraints only. If there are any signal-specific constraints to be added, such as specifying constant, static, or mutually exclusive signals, this can also easily be achieved in the setup phase. Once setup is completed, clock domains and crossings can be analyzed.

Comprehensive Structural Checks

The JasperGold CDC App automatically identifies the vast majority of CDC synchronizers in use today, including nDFF, MUX, FIFO structures, and more. On the rare occasion when a synchronizer type is not automatically identified, user-defined synchronizers can be easily identified to the tool. As well as checking that crossings are correctly protected, the tool analyses convergence and reconvergence paths. There are realistic default values for the cycle depth of these analyses, which can be readily modified by the user if necessary. These checks are illustrated in Figure 2.

Functional Checks Powered by Formal Technology

In addition to the structural checks, the JasperGold CDC App provides powerful functional checks, highlighted by the green glow in Figure 2. It is common for FIFO synchronizers to use a gray code protocol, and other synchronizer types may use handshake and other protocols. Structural checks alone do not ensure these protocols are always observed. Formal-powered functional checks do, and regularly find bugs that will either be found much later, or worse, are common bug escapes to silicon. These property-based formal checks are automatically created by the JasperGold CDC App and require no specialist formal knowledge from the user. For example, checks are automatically generated to ensure that data in the source domain is stable for at least one sampling edge of the destination register. All relevant control and clocking signals are considered to avoiding spurious counter examples and improve convergence.

Metastability Modeling and Injection

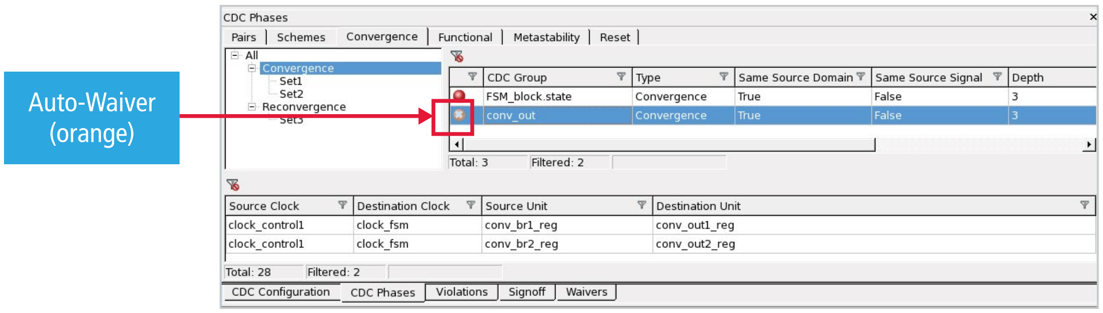
The JasperGold CDC App provides a metastability injection flow. Even when CDC synchronizers are used correctly, and especially with simpler synchronizer types, there are potential functional problems that can be caused by either glitches or cycle delays on signals. The JasperGold CDC App provides modeling for these metastability effects, plus an injection flow to see whether these glitches can cause functional problems in the design. This capability is especially useful to catch issues due to reconvergence paths. The metastability effect can be injected formally in the JasperGold CDC App, or can also be exported to simulation. In the latter case, there is a tight integration with Cadence's Xcelium™ Parallel Simulator. This metastability flow can also handle combinational logic on the CDC path.

Reset Domain Crossing Checks

The combination of reset signals crossing clock domains and the use of synchronous resets, or different asynchronous resets in the same clock domain, can cause incorrect and unexpected behaviors during reset and post-reset design bring-up. The JasperGold CDC App provides specific checks to identify such reset domain crossing (RDC) problems.

Formal Intelligence to Reduce Noise

As the only CDC solution based on a technology-leading formal verification platform, there are many ways the JasperGold CDC App applies formal intelligence to improve user experience. The most common user complaint we hear regarding other solutions is "noise". This refers to presenting too many violations to the user in an unstructured way, and continually presenting the same violations at different stages of the analysis. A true formal CDC solution such as the JasperGold CDC App uses intelligent root-cause analysis, and various methods of grouping and filtering similar violations, to present violations to the designer in a much more user-friendly way. The JasperGold CDC App also uses formal-powered Visualize™ debug technology for finding the cause of the violations. But the JasperGold CDC App can do even more. As illustrated in Figure 4, the JasperGold CDC App has an auto/safe waiver capability, so it can automatically waive violations based on user-specified structural information.



CDC Group	Type	Same Source Domain	Same Source Signal	Depth
FSM_block.state	Convergence	True	False	3
conv_out	Convergence	True	False	3

Total: 3 Filtered: 2

Source Clock	Destination Clock	Source Unit	Destination Unit
clock_control1	clock_fsm	conv_br1_reg	conv_out1_reg
clock_control1	clock_fsm	conv_br2_reg	conv_out2_reg

Total: 28 Filtered: 2

Figure 3: Auto/safe waiver capability in the JasperGold CDC App

The auto/safe waiver capability uses formal technology to guarantee a safe waiver, by allowing the user to add a condition upon which the waiver will be effective. This is especially important for IP reuse, where those conditions, usually made on the boundaries of module, can be checked at the top level. For example, a certain violation may only be an issue if a quasi-static signal's value is not actually static. The JasperGold CDC App generates a property-based functional check to validate the assumption of a static value. If the functional check passes, the auto-waiver is promoted as a safe waiver. Otherwise, the auto-waiver status is reverted to a violation.

RTL Signoff and Hierarchical Analysis

The JasperGold CDC App is part of an overall RTL signoff strategy where the momentum now is to move all possible checks to the RTL design and IP levels. This trend has emerged in part because IP blocks are often reused in many system-on-chip (SoC) designs. At the same time, particularly with larger organizations, IP and SoC design teams are often separate. So, it has become pragmatic to perform checks early on. The combination of robust IP and a good methodology can help minimize late surprises and encourage much easier netlist signoff checks.

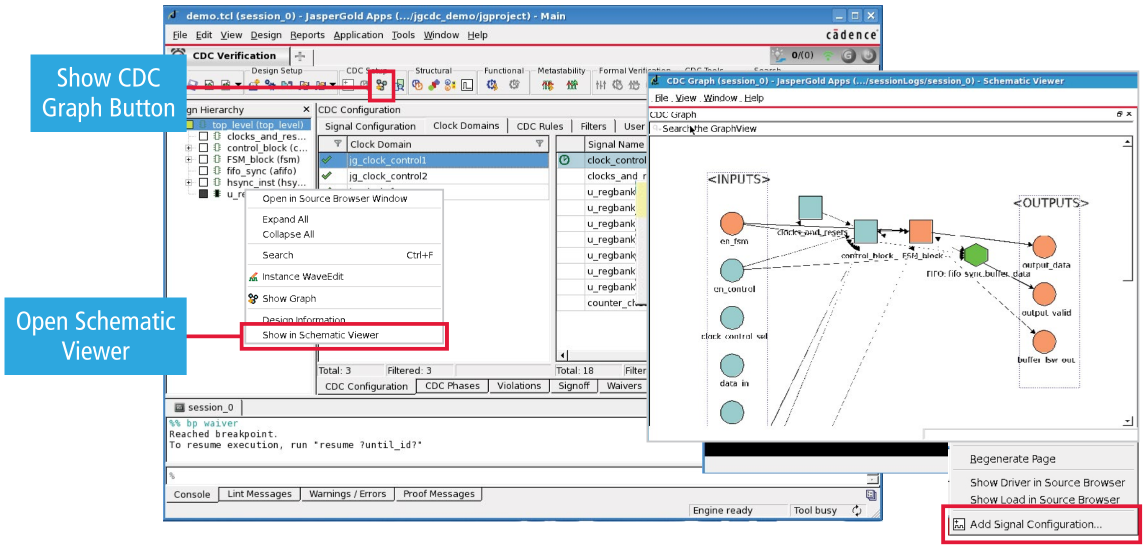
On the RTL design side, the goals toward achieving robust, reusable IP include:

- Performing lint, DFT, and automatic formal checks
- Performing checks to assure CDC-clean RTL
- Easy debug and waiver handling

The primary goal of RTL implementation checks is to achieve SoC integration signoff via:

- Signoff checks with SoC-specific constraints
- Full chip-level capacity and performance
- Reuse of IP-level metadata/waivers to reduce noise

An integrated RTL signoff flow, therefore, would involve debugging and fixing RTL at the source. Then, as signed-off blocks are integrated together to assemble the RTL for the complete chip, waivers are preserved, and hierarchical analysis can be performed at chip level. At this stage, by default, the user only sees new violations occurring because of the integration, including chip-level constraints and different clock specifications. The user can choose to see all violations if desired, but to display violations previously waived is usually regarded as "noise." Then, any issues in SoC integration or implementation should be resolvable without requiring RTL changes to the reused IPs.


Show CDC Graph Button
Open Schematic Viewer
demo.tcl (session_0) - JasperGold Apps (.../jgcdc_demo/jgproject) - Main
File Edit View Design Reports Application Tools Window Help
CDC Verification
Open in Source Browser Window
Expand All
Collapse All
Search Ctrl+F
Instance WaveEdit
Show Graph
Design Information
Show in Schematic Viewer
CDC Graph (session_0) - JasperGold Apps (.../sessionLogs/session_0) - Schematic Viewer
<INPUTS>
<OUTPUTS>
Regenerate Page
Show Driver in Source Browser
Show Load in Source Browser
Add Signal Configuration...
Console
Lint Messages
Warnings / Errors
Proof Messages
Engine ready
Tool busy

```
%% bp waiver
Reached breakpoint.
To resume execution, run "resume ?until_id?"
```

Figure 4: JasperGold Visualize Interactive Debug Environment with graph view

Competitive solutions are considered to be noisy—violation alerts continue to pop up even after the issue has been resolved. With the JasperGold CDC App, once a violation has been addressed and turned either into a waiver or resolved, it can be filtered out. In this environment, you can generate reports and analyze violation messages, use advanced waiver capabilities, and more. The solution works with the JasperGold Visualize™ Interactive Debug Environment in which you can create formal traces to debug, and interact with them without executing the design. You can also link back to the source line in the source code that is triggering the violation. Additionally, as shown in Figure 4, there is a CDC-specific graph view to abstract the important elements of the clock crossings under analysis, fully integrated with the source code, violations, and property views, for an improved intuitive debug user experience.

Summary

The need to minimize dynamic power is a strong factor driving an explosion of the number of clock domains in today's designs. Additionally, the need to design robust RTL IP that can be reused in many different chip implementations is accelerating the need to fully check for CDC issues at the RTL design stage. The JasperGold CDC App brings true formal intelligence to add innovative functional checks and formal-powered waiver handling to state-of-the-art structural checks, to assure CDC-clean RTL designs, and a much-improved reduced-noise user experience.

Further Information

Learn more about the JasperGold CDC App at https://www.cadence.com/content/cadence-www/global/en_US/home/tools/system-design-and-verification/formal-and-static-verification/jasper-gold-verification-platform/jaspergold-clock-domain-crossing-app.html.