

Challenges in Verification of Clock Domain Crossings

Vishnu C. Vimjam and Al Joseph
Real Intent Inc., Sunnyvale, CA, USA

Notice of Copyright

This material is protected under the copyright laws of the U.S. and other countries and any uses not in conformity with the copyright laws are prohibited. Copyright for this document is held by the creator — authors and sponsoring organizations — of the material, all rights reserved.

DESIGN
AUTOMATION
CONFERENCE

WHITE PAPER: Verification

Challenges in Verification of Clock Domain Crossings

Vishnu C. Vimjam and Al Joseph

Real Intent Inc., Sunnyvale, CA, USA

Abstract— Emerging systems have three dimensions of complexity when it comes to making them safe with respect to clock domain crossings (CDCs). First, the number of asynchronous clock domains can range from several tens to well over a hundred for complex systems with many components. Second, the master clock frequencies vary per component. It is not uncommon for the ratio between the fastest and the slowest clock frequencies to be greater than 10. Third, the clock frequencies themselves can change dynamically during the course of operation of the chip (for example, when switched from one mode to another to save power). As a result, CDC verification becomes critical to ensure that metastability is not introduced into the design. In this article, we describe several situations and provide examples to showcase challenges in CDC verification.

Index Terms— Clock domain crossings, asynchronous FIFOs, metastability, synchronizers.

I. INTRODUCTION

Today's systems have a multitude of components working with different clock domains running at varying speeds. Examples range from the small mobile phone chips to huge graphics chips or microprocessors that interface with a variety of buses and I/Os [1,2]. Signals that cross clock-domain boundaries – commonly referred to as *clock domain crossings*, or CDCs – are typically classified into two types, (i) synchronous and (ii) asynchronous. Synchronous crossings are those for which the receiving domain has a phase/frequency relationship with the sending domain. These crossings are timed and verified robustly. On the other hand, asynchronous crossings are those for which there is no relationship between the sending and receiving clocks. These clocks originate from different clock generators, or from their derivatives. As a result, timing cannot be accurately verified since the order of clock edges cannot be guaranteed.

Consider the example in Figure 1(a), where a signal “SA” flows from the ClkA domain to the ClkB domain. Without loss of generality, assume FlopB is active with respect to the posedge of ClkB. Since the order of edges of ClkA and ClkB cannot be guaranteed, SA needs to be stable by the time the corresponding edge of ClkB arrives such that SA can be correctly captured in FlopB. If not, we get into the undefined behavior called *metastability*. As shown in Figure 1(b), if a 0->1 transition on SA falls within the setup/hold times of FlopB, it can be either be treated as 0, or it can be treated as 1, or the flop FlopB can reach an intermittent metastable value [1,2]. Such a state is called a “metastable state”; its occurrence injects metastability into the downstream logic, leading to undefined or incorrect behavior. An added complexity is that the side effects of metastability might not be seen immediately, but only after being propagated for some clock cycles. Such situations are hard to debug and can consume a significant fraction of verification time. This issue of metastability can manifest itself in a variety of design styles and structures. Ensuring that CDCs function reliably requires very strict handshaking or FIFO-based synchronization schemes that work across the entire range of operating frequencies for the clocks.

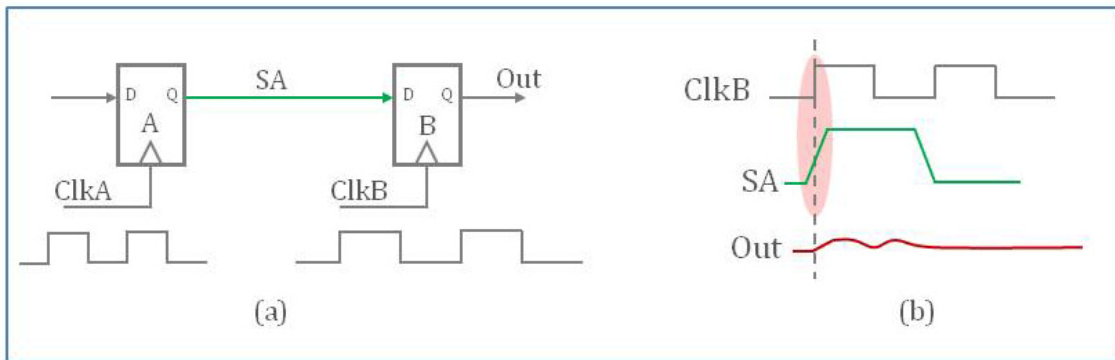


Figure 1: Metastability due to asynchronous crossings.

Typical usages of signals flowing asynchronously from one domain ClkA to another domain ClkB are as follows:

1. Reset signals
2. Single-bit data signals
3. Multi-bit data signals
4. Synchronized control signals

Reset signals are used to reset the logic in the receiving domain during chip-reset phase or whenever interrupts such as software resets or aborts occur in the sending domain. Single-bit data signals are typically used to convey some sort of status to the receiving domain. For example, single-bit data signals can be used to convey that the sending domain is busy or to gate the receiving domain clock, etc. Both of these kinds of signals need to be synchronized (usually with a double-flop synchronizer [3]) before usage in the receiving domain. Synchronization is sufficient because these signals are intended to transition intermittently and stay stable for most of the time.

On the other hand, multi-bit data transfer is used to transfer buses of data signals between domains. If each of these bits is individually synchronized, the outputs of the synchronizers lose their correlation due to the metastability problem. Hence, some sort of common control mechanism is required to (i) let the receiving domain know that the transmitted multi-bit data is valid and (ii) let the receiving domain capture that data only when it is stable. This is often accomplished via handshake-based synchronized control signals or via FIFO-based synchronization [4,5].

II. CDC VERIFICATION CHALLENGES

Early solutions for CDC analysis only verified single-bit metastability management (synchronizers implemented as back-to-back flops) and data-bus metastability management (controlled by a synchronized common enable signal). Designers' understanding of CDC analysis, even up to this day when CDC analysis has become mission-critical for SOC designs, is that it only requires checking these two attributes.

In reality, the above two checks are just the tip of the CDC analysis iceberg. To begin with, they represent only limited checking of metastability management. Analysis must be performed to verify cycle jitter tolerance in data crossings, cycle jitter in control crossings, data corruption in fast to slow clock crossings (i.e., pulse width causing missing data), Glitch issues with clock gating implementation (see II.E below), reconvergence of signals synchronized separately from a single clock domain (see II.A below), correct implementation of asynchronous FIFO protocols, and correct implementation of resets that cross multiple domains. Further, all of these must be verified assuming the presence of metastability in the design, otherwise the verification is not realistic. For example, an underflow check for a FIFO (read-before-write) might be functionally proven when read and write are synchronous, but the same FIFO implementation might not work when the read and write sides are asynchronous. Because of transformations induced by timing-driven synthesis optimizations, as well as test-driven and power optimization-driven modifications of the clock structures, performing CDC verification at the netlist level is a must.

Verifying the above issues requires a *full-custom* combination of structural analysis, simulation-based techniques and static formal property checking, *and must fully account for occurrences of metastability in the design*. Note that none of these approaches can independently verify all types of issues, but their benefits can be leveraged to achieve a comprehensive and accurate CDC verification solution.

The following subsections show several common scenarios that designers often overlook, potentially leading to expensive re-spins due to CDC failures.

A. Missing Gray Encoding

Consider the crossings shown in Figure 2. Three synchronized control signals flow from a slower clk1 domain to a faster clk2 domain, then reconverge in the clk2 domain and enter an FSM. Since the multiple control signals are not Gray-coded, more than one sending flop can transition

simultaneously (for example, 000->011). When these signals reach the clk2 domain, they can be captured in an invalid state (for example, 010) which is neither the previous nor the current state. In typical designs, when there are multiple sending domains, the hand-shaking protocols generally ensure that they are mutex-constrained; hence, Gray encoding is not necessary. On the other hand, missing Gray coding in situations such as that of Figure 2 can cause unwanted behavior. Such situations can be verified by modeling metastability and proving, via formal analysis or simulation, that there are no side effects.

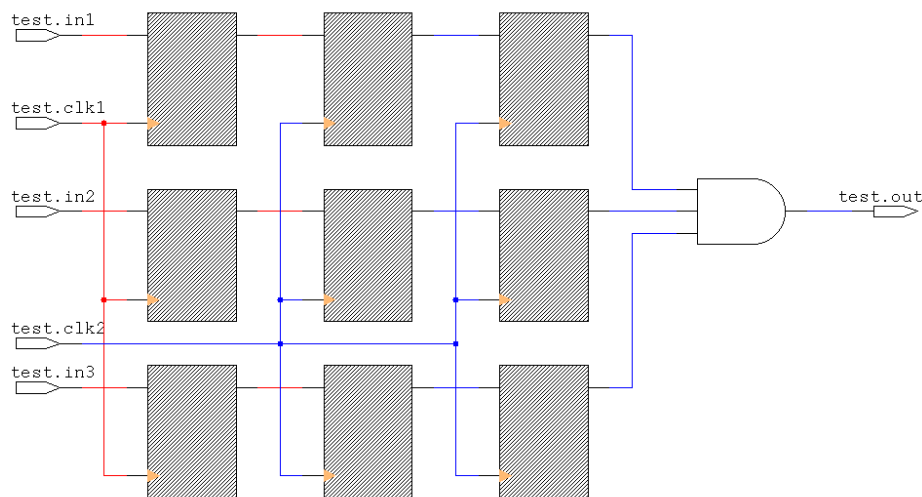


Figure 2: Reconvergence of synchronized control signals.

B. Global Loss of Data Correlation

Consider the design shown in Figure 3. Data from an originating clock domain clk1 is synchronized and transferred to two different domains clk2 and clk3. After operations are performed on this data, it is passed back to clk1 domain, where additional analysis is done. This kind of transfer has to be designed very carefully, otherwise correlation among the data buses can be easily lost in the transfers, leading to chaos at the outputs. Virtually the only way to verify these will be to write user assertions and make sure they pass in the presence of metastability.

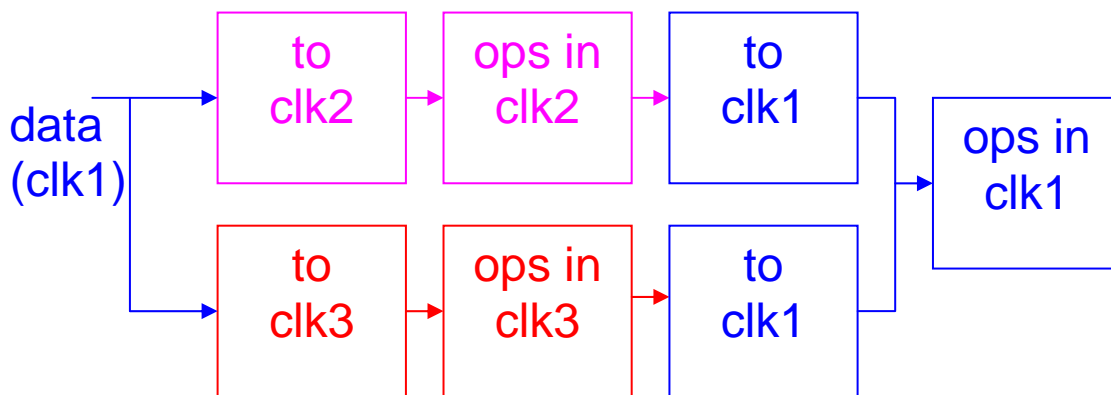


Figure 3: Transfer of data across multiple domains leading to global correlation issues.

C. Loss/Corruption of Data

Consider the design shown in Figure 4. A data transfer (the second input of the mux) goes from clk1 to clk2, and is controlled by a synchronized control signal. While it is important that the sending domain assert a control signal signaling a transfer of data, it is equally important that this signal is received correctly by the receiving domain. If the asserted pulse on the control signal is not large enough for it to be captured in the receive domain, the data from the sending domain might be missed or misread. This typically happens due to an incorrect handshake mechanism or when clk1 is much faster than clk2. Employing a combination of structural and formal techniques in the presence of metastability will be the only robust way to verify such situations.

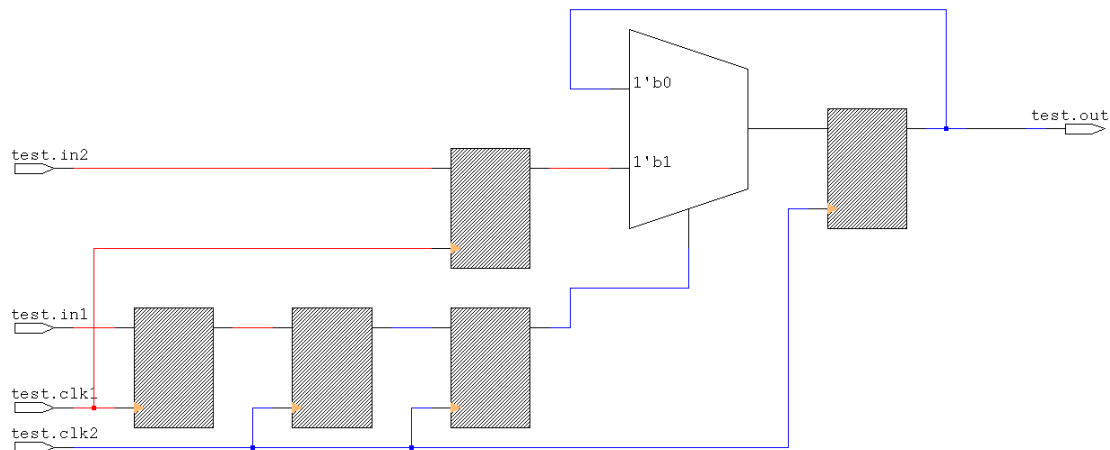


Figure 4: A synchronizer being used to control a data transfer.

D. Clock-Gating Uncertainty

Consider the design shown in Figure 5. Due to power optimization constraints, synthesis has determined to gate the clk2 domain under certain conditions arising from the clk1 domain. Designers merrily sign off once equivalence checking has passed, only to realize later that these gating signals are not synchronized – causing expensive ECOs. Lack of synchronization can lead to uncertainty and glitches along clock paths that can corrupt data for many flops. Employing structural methods and leveraging the benefits of simulation testbenches can help identify such violations.

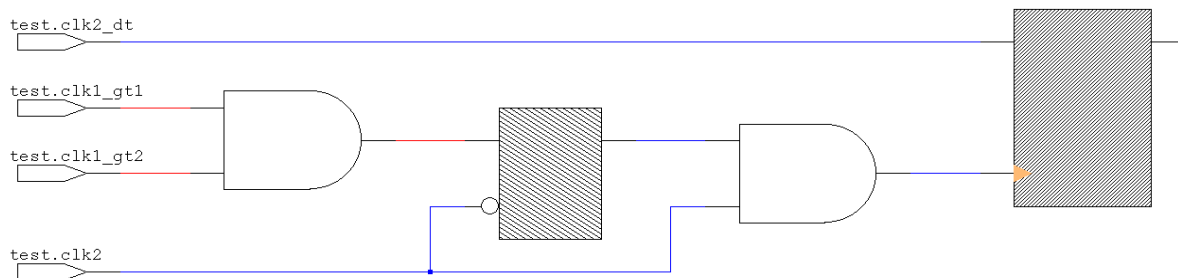


Figure 5: Clock gating from an asynchronous source.

E. Glitches on Netlists

Depending on the types of transformations/optimizations made during logic synthesis, the resulting netlists are likely to have glitches. While these do not pose a problem with timed, synchronous crossings, they can cause serious damage for asynchronous ones. Consider the simple schematic shown in Figure 6. Combinational logic from the clk1 domain is synchronized and used as a control signal in the clk2 domain. Glitches arising in this logic can propagate to the first synchronizer flop and can be latched into the destination domain, resulting in corrupt data being captured.

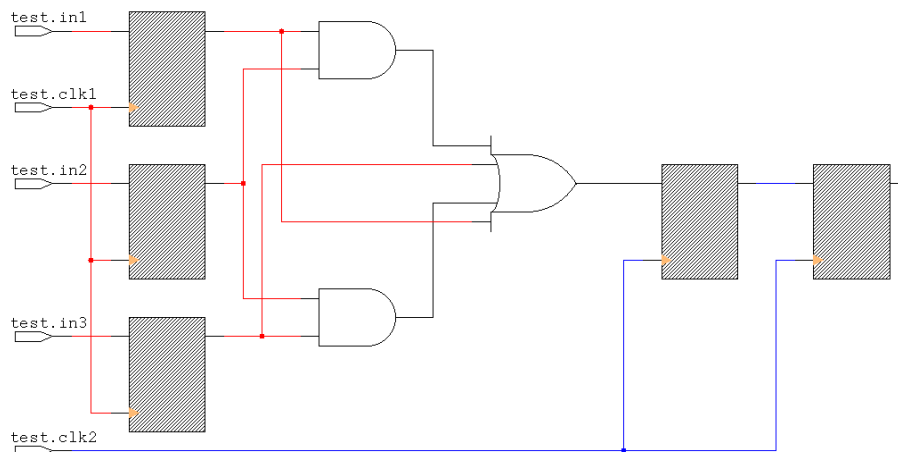


Figure 6: Combinational logic along crossing paths.

F. Dynamic Frequency-Changing Clocks

Consider the schematic shown in Figure 7. There are two clocks, clk1 (blue) and clk2 (red). For simplicity, we assume that all the flops in the design are posedge triggered. A data transfer (shown as DATA) between the clk1 and the clk2 domains is controlled by the 3-flop toggle synchronizer (shown as SYNC). The toggle synchronizer has two flops that synchronize the control signal and a third flop to detect that a valid request signal is sent from the clk1 domain. Notice that the output of the second flop is fed back into the clk1 domain (shown as FEEDBACK) to acknowledge that the DATA has been received. Once the FEEDBACK is seen, the clk1 domain sends new DATA across the interface.

First, let us consider the case when the clk1 domain is faster than the clk2 domain by up to a factor of 2. Since they are relatively asynchronous, there could be a maximum of 2 posedges of clk1 within any given cycle of clk2. With formal analysis, we can determine that the above feedback structure is sufficient for the CDC interface to work correctly (irrespective of the duty cycles and transition times of the clocks). This is because by the time the clk1 domain receives the feedback signal, clk2 would have received the data.

Now consider the case when the clk1 domain is faster than the clk2 domain by a factor of 3. In other words, there can be 3 posedges of clk1 within a given cycle of clk2. In this case, the clk1 domain receives the feedback signal before the clk2 has received the data. Then clk1 can immediately proceed to transfer new data, which can corrupt the old data and/or cause metastability. The probability of failure increases rapidly as the clk1 frequency increases. The interface can be made frequency-independent by taking the feedback signal from the third flop of the toggle synchronizer instead of the second flop; this guarantees that the clk2 domain receives data prior to sending a feedback.

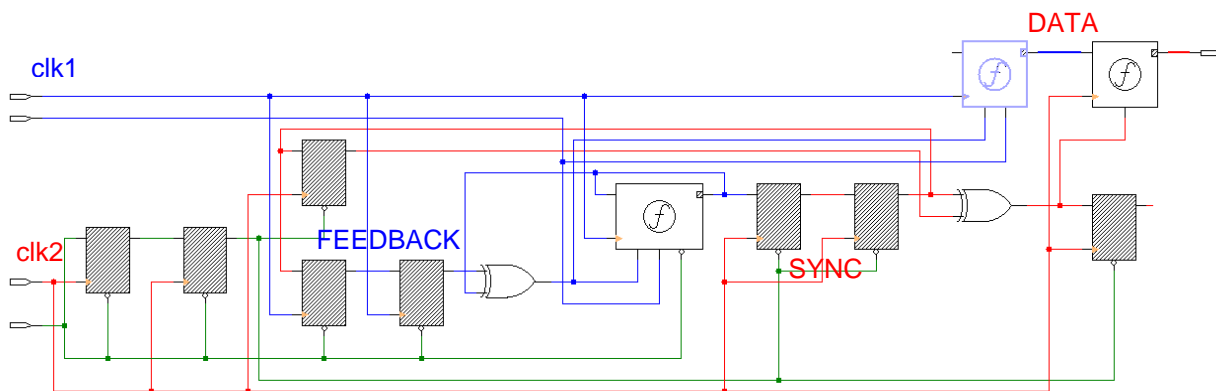


Figure 7: Incorrect feedback mechanism.

Without the fix, the CDC interface in the example clearly doesn't work over the entire operating range of frequencies. Predictability becomes even worse when the clock frequencies are changed dynamically during the course of the chip operation (this is done in practice by applying appropriate software resets, etc. carefully between the operating modes).

III. CONCLUSION

As asynchronous crossings become more mainstream because of larger dies and greater system-level complexity on chip, it is clear that managing metastability is of paramount importance. The analysis of designs for proper metastability management has come to be known as CDC analysis. Clock domain crossings are an inevitable aspect of modern designs and will only increase as more features are added on SOCs. CDC verification is essential to ensure that users do not miss any corner cases or end up spending significant time resolving issues.

As with metastability verification, all of the above issues are very difficult to characterize and verify with simulation-based techniques. There have been many silicon re-spins as a result of not comprehensively verifying the above issues. While there is no straightforward solution to these issues, user can apply a variety of approaches, ranging from structural analysis to leveraging testbench simulation in the presence of metastability to full-scale formal verification. It is also imperative that CDC analysis is done at the netlist level in addition to the RT level. Automated CDC tools such as Meridian [6] from Real Intent Inc. have pioneered the solutions and the technology required to verify that a design conforms to CDC intent. Meridian was designed from the ground up with a first-principles understanding of CDC failure modes. Its complete asynchronous-control-structure recognition technology provides a viable signoff-quality, low-noise solution with easy setup and report-analysis capabilities, comprehensive checking, and practical execution times.

REFERENCES

- [1] R. H. Parker, "Caution: Clock Crossing, A Prescription for Uncontaminated Data Across Clock Domains", <http://chipdesignmag.com/display.php?articleId=32&issueId=5>
- [2] P. Narain and C. Cummings, "Clock Domain Crossing Demystified: The Second Generation Solution for CDC Verification", Real Intent CDC White Paper, <http://www.realintent.com/real-intent-products/white-papers>
- [3] R. Ginosar, "Fourteen Ways to Fool Your Synchronizer", *Proc. Ninth IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, 2003, p. 89.
- [4] C. Cummings, "Simulation and Synthesis techniques for Asynchronous FIFO design", *Synopsys Users Group Conference (SNUG) User Papers*, March 2002. Also available at www.sunburst-design.com/papers
- [5] C. Cummings and P. Alfke, "Simulation and Synthesis Techniques for Asynchronous FIFO Design With Asynchronous Pointer Comparisons", *Synopsys Users Group Conference (SNUG) User Papers*, March 2002. Also available at www.sunburst-design.com/papers
- [6] <http://www.realintent.com/real-intent-products/meridian>