TECHNICAL PAPER

cadence

CLOCK DOMAIN CROSSING

CLOSING THE LOOP ON CLOCK DOMAIN FUNCTIONAL IMPLEMENTATION PROBLEMS

TABLE OF CONTENTS

1 Overview 1

2 CDC basics 1

3 Structural design for synchronization (*s*CDC) 3

4 Data stability (*f*CDC) 6

5 A complete CDC solution 7

6 EDA tools 10

7 Conclusion 11

8 References 12

TABLE OF FIGURES

Figure 1 Single clock domain 2

Figure 2 The CDC path 2

Figure 3 Metastability basics 2

Figure 4 Two flip-flop synchronizer solution 3

Figure 5 Control path and data path synchronization 3

Figure 6 Convergence issue 4

Figure 7 Divergent crossover path issue 5

Figure 8 Fanout of metastable signal 5

Figure 9 Effects of fanout of metastable signal 5

Figure 10 Reconvergence and waveform 6

Figure 11 200MHz signal synced into a 166MHz domain 6

Figure 12 Timing-closure–only methodology 7

Figure 13 Methodology with CDC verification 8

Figure 14 *f*CDC checks and their locations 9

Figure 15 Handshake check 9

Figure 16 Encounter Conformal CDC detects structural design issues with synchronization 11

1 OVERVIEW

Shrinking device sizes and increasingly complex designs have created multimillion-transistor systems running with multiple asynchronous clocks with frequencies as high as multiple gigahertz. SoC systems have multiple interfaces, some using standards with very different clock frequencies. Several modern serial interfaces are inherently asynchronous from the rest of the chip. There is also a trend toward designing major sub-blocks of SoCs to run on independent clocks to ease the problem of clock skew across large chips.

Design methodologies have traditionally focused on partition-based implementation and verification. Often these partitions are based on clock domains. The cross-clock domain crossing (CDC) signals pose a unique and challenging issue for verification. Traditional functional simulation is inadequate to verify clock domain crossings. While static timing analysis (STA) is an integral part of the timing closure solution, little attention has been paid to addressing proper clock domain implementation and verification. Existing methods provide an ad hoc partial verification that is manual, time consuming, and error prone.

If the sources of potential errors are not addressed and verified early on, designs can end up with functional errors that are only detected late in the design cycle — or even worse, during post-silicon verification. The cost of fixing errors at this stage is enormous. We know of large system houses in which chips are "dead in the water" due to CDC problems.

Several errors could be caused by cross-CDC signals:

- **Structural issues (*s*CDC):** If the data input to a storage element changes too close to a clock edge, the element may go into a metastable state and the output cannot be reliably used. Asynchronous clock domain crossings are particularly prone to metastability failures. To address this issue, the circuit must be designed to "buy time" so the metastable signal can settle to a stable value, typically using synchronizers.

  After completing the synchronization, the structures beyond the synchronizers still matter. For example, the design must ensure that the synchronized signals do not converge. Reconvergence can create functional errors.

- **Functional errors (*f*CDC):** Designers must ensure that the stability and functionality on either side of the CDC circuit are handed over properly. Otherwise, there could be loss of signal values for signals passing between clock domains, with data instability in the receiving clock domain.

Only automatic formal verification techniques can ensure that multiclock designs are correct prior to tapeout. The CDC verification solution must address this verification challenge, while maximizing overall productivity and effectiveness. The CDC solution needs to cover clock domain analysis and structural and functional verification, addressing both register-transfer–level (RTL) and gate-level verification needs. Utilizing the Cadence® CDC solution enables development teams to reduce the overall verification effort and lessen the risk of costly re-spins due to asynchronous clock domain crossing errors.

2 CDC BASICS

The design issues and challenges of handling the signal crossing domains will be discussed later in this paper. First, let's look at some CDC basics.

2.1 CLOCK DOMAINS

A clock domain is defined as that part of the design driven by either a single clock or clocks that have constant phase relationships. A clock and its inverted clock or its derived divide-by-two clocks are considered a clock domain (synchronous). Conversely, domains that have clocks with variable phase and time relationships are considered different clock domains.

In *Figure 1*, the design has a single clock domain because the divCLK is the derived divide-by-two clock of the master clock CLK.

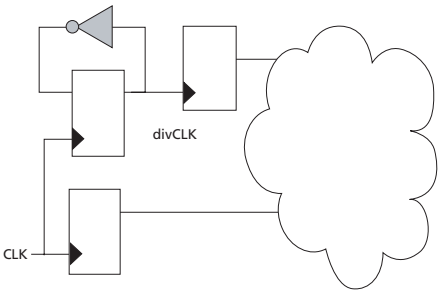

Figure 1: Single clock domain

In *Figure 2,* multiple clocks come from different sources. The sections of logic elements driven by these clocks are called clock domains, and the signals that interface between these asynchronous clock domains are called the clock domain crossing (CDC) paths. The DA signal is considered an asynchronous signal into the clock domain — no constant phase and time relationship exists between CLK A and CLK B.

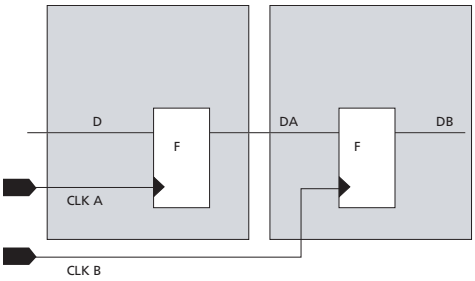

Figure 2: The CDC path

2.2 ASYNCHRONOUS SIGNAL BASICS

Asynchronous signals, including CDC signals, have the potential to become metastable.

2.2.1 Metastability

The proper operation of a clocked flip-flop depends on the input being stable for a certain period of time before and after the clock edge. If the setup and hold-time requirements are met, the correct output will appear at a valid output level (either VoL or VoH) at the flip-flop output after a maximum delay of *t*CO (the clock-to-output delay). However, if these setup and hold-time requirements are not met, the output of the flip-flop may take much longer than *t*CO to reach a valid logic level. This is called unstable behavior, or metastability.

As *Figure 3* illustrates, if CLK B samples DA while DA is changing (at the rising edge of CLK and falling edge of D), then DB will be metastable.

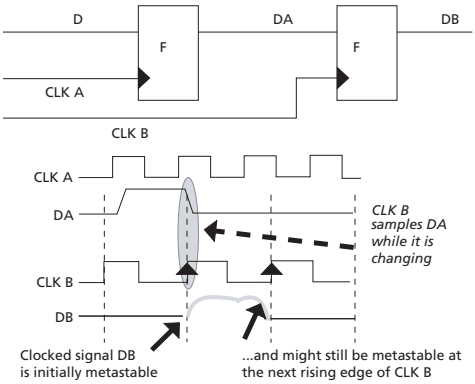

Figure 3: Metastability basics

Metastability cannot be avoided, but a solution for handling the metastable signal enables proper functioning of the design.

2.2.2 Mean time between failures

The metastability occurrences can be predicted by using the mean time between failures (MTBF) formula:

$$MTBF = \frac{e^{C2*tMET}}{C1 * fclk * fdata}$$

Where *C1* and *C2* are constants that depend on the technology used to build the flip-flop; *tMET* is the duration of the metastable output; and *fclk* and *fdata* are the frequencies of the synchronous clock and the asynchronous input, respectively.

3 STRUCTURAL DESIGN FOR SYNCHRONIZATION (*s*CDC)

3.1 SYNCHRONIZATION (CONTROL PATHS)

Designers can use special metastable hardened flops for increasing the MTBF. For example, in *Figure 4,* a synchronizer flop is used following the signal DB. So, instead of the metastable signal DB being used in the function downstream as in *Figure 3,* the stable signal DB2 is used in the logic downstream.

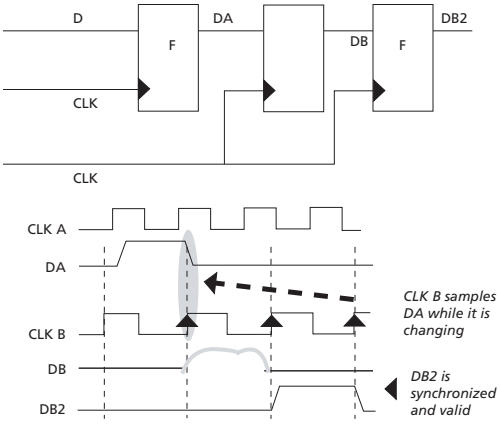

Figure 4: Two flip-flop synchronizer solution

3.2 SYNCHRONIZATION (CONTROL AND DATA PATHS)

Designs typically have control paths and data paths. As shown in *Figure 5,* the control signals are usually flop-synchronized while the data paths use the synced-in control signals to synchronize the data path. The data path uses a controlled synchronizer MUX to do the domain crossing. This control MUX is sometimes called D-MUX, MUX synchronizer, or sync MUX. We use the term *MUX synchronizer*, which is a synonym for MUX synchronization scheme.

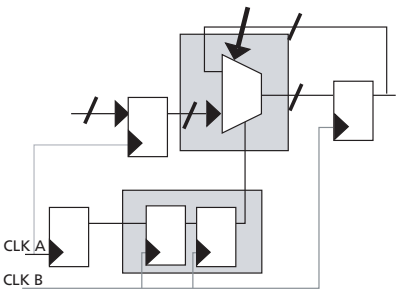

Figure 5: Control path and data path synchronization

The MUX synchronizer has a critical requirement for all input in terms of the domains and functionality:

- The select input of the MUX comes from the *destination* domain (domain into which the signal is being synchronized)
- One of the MUX inputs is coming from the destination domain — that is, the holding loop
- The MUX inputs can be source, destination, or user-specified static signals
- The logic between the MUX synchronizer and the destination flop is driven by the destination domain or static signals

3.3 CONVERGENCE IN THE CROSSOVER PATH

Clock domain crossover paths are false paths for timing tools; any logic in this path must be carefully crafted and verified, because the logic can cause glitches and create functional errors downstream.

In *Figure 6*, although the two source flops give the pulse at the same time, the propagation delay (T_d) in post-layout masks out the pulse. Since it is a false path (ignored by the timing tool), the design techniques should consider these occurrences.

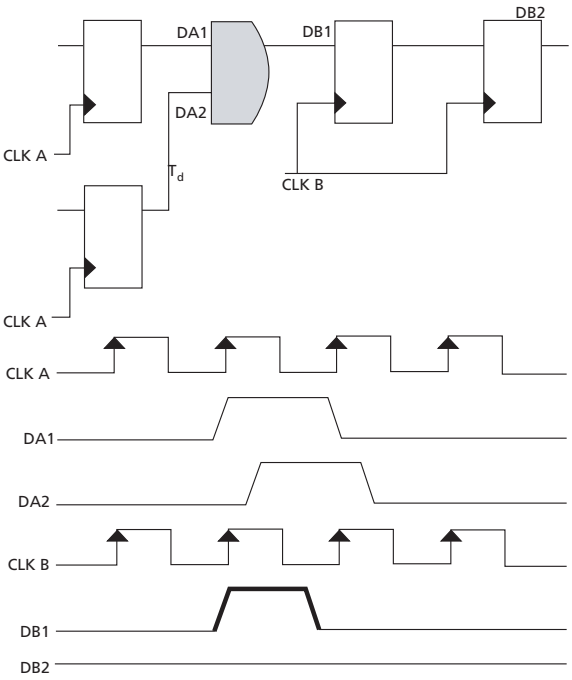

Figure 6: Convergence issue

3.4 DIVERGENCE IN THE CROSSOVER

A divergent logic style to multiple synchronization paths runs the risk of causing functional errors. As *Figure 7* illustrates, due to the propagation delay and different metastable settling times, the Fsm1_en and Fsm2_en could start at different times. This type of structure should be avoided by fanning out a single FSM enable after synchronization to the two FSMs.

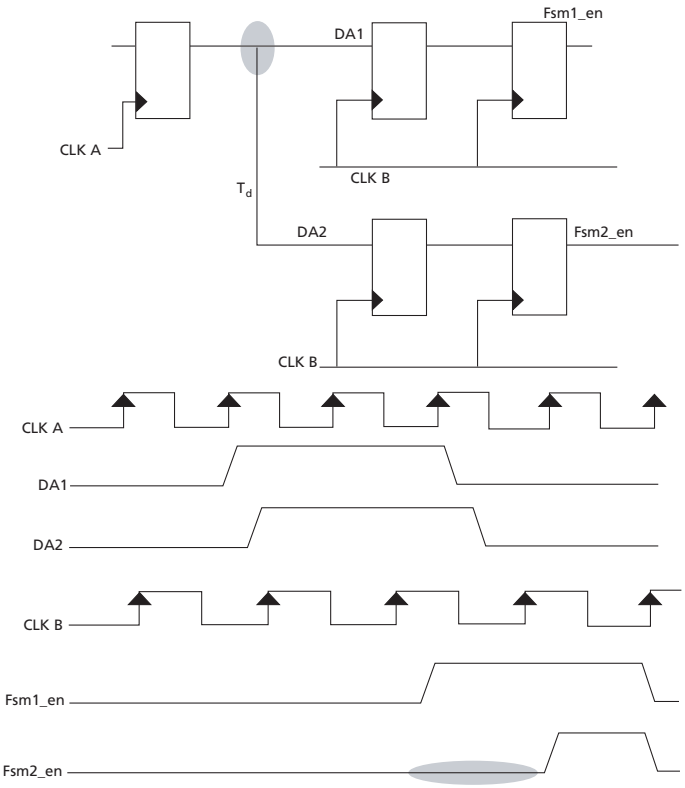

Figure 7: Divergent crossover path issue

3.5 DIVERGENCE OF METASTABLE SIGNALS

Metastable signals are unstable signals, and circuit designs must be put in place to squelch this behavior. The usual style is to loop back to the generating flop—that is, the MUX loop (as shown in *Figure 8*).

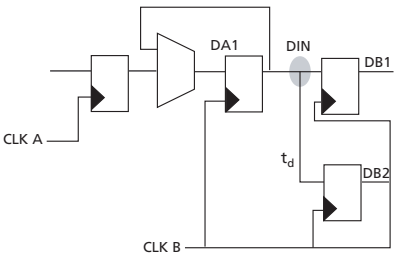

Figure 8: Fanout of metastable signal

A divergence in the metastable signal can cause functional errors, as seen in *Figure 9*: DB1 and DB2 came out at two different clock edges as the settling and latching values of these metastable signals to the two flops are at different times.

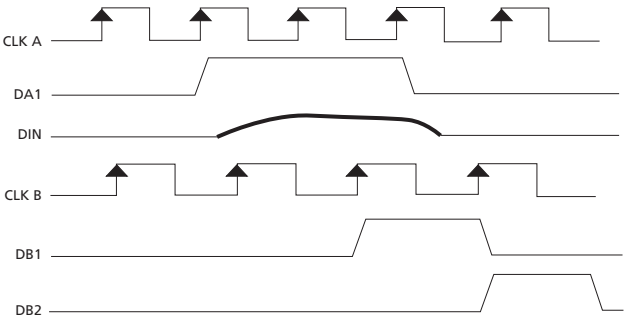

Figure 9: Effects of fanout of metastable signal

3.6 RECONVERGENCE OF SYNCHRONIZED SIGNALS

Once synchronization is completed, the structures beyond the synchronizers still matter. The design must ensure that the synchronized signals do not converge—reconvergence can create functional errors. In *Figure 10*, the post-synchronization logic can cause a glitch in the signal DB.

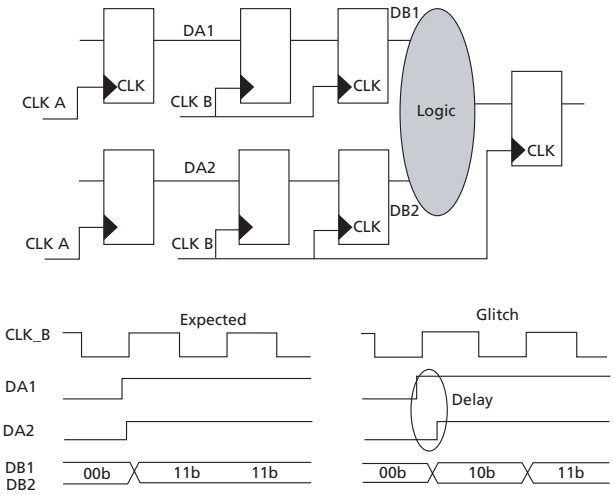

Figure 10: Reconvergence and waveform

4 DATA STABILITY (*f*CDC)

Up to this point, we have discussed the structural issues and styles needed to have proper synchronization across two asynchronous clock domains. The functionality of the holding and latching circuit across the CDC path should ensure the proper transfer of the signal. The holding of the source register and the latching of the destination register across the CDC path must be properly designed. Otherwise, issues such as shown in *Figure 11* (data loss) affect the proper functioning of the circuit. This problem is prevalent when a fast clocked signal is synced into a slow clocked signal. The functionality should be built into the circuits for such CDC paths.

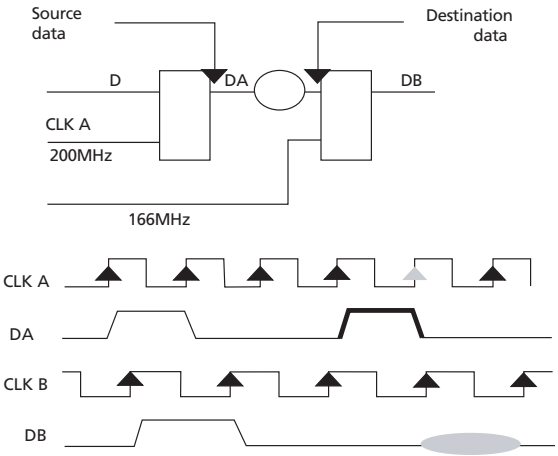

Figure 11: 200MHz signal synced into a 166MHz domain

4.1 EXAMPLE OF *f*CDC

When a signal from a fast clock domain crosses over to a slow clock domain, design aspects must ensure that the signal from the fast clock domain is held long enough to be captured properly by the destination domain. There could also be a scenario in which phase-shifted signals of a slow clock are interacting, and the resulting signal's clock changes every phase. This situation would not be captured by the destination clock, resulting in loss of data.

In *Figure 11*, a 200MHz signal is being synced into a 166MHz domain. The source signal DA must be held for at least two clocks to ensure a capture edge will always fall in this width. This is not the case in *Figure 10,* however, which shows a capture once, but not the second time (DB). The capture edge is asynchronous and can arrive at any time of the source signal. If the DA signal were stretched to two cycles, there would definitely be an edge for CLK B to capture the signal.

4.2 GRAY ENCODING

A problem for multiclock designs is loss of the relationship between signals—either bits of a bus or independent signals — when crossing clock domains, which can cause unpredictable results.

For multibit signals such as buses, the usual solution is to use a Gray code when crossing a clock domain boundary. A Gray code ensures that only a single bit changes as the bus counts up or down. This makes it possible to detect whether all bits have been captured together by the receiving clock or if they have been skewed across multiple clock cycles due to metastability or differing delays among the bus bits.

For example, in an asynchronous FIFO, the read and write pointers cross over to the write and read clock domains, respectively. These pointers are control signals that are flop-synchronized. The signals are Gray-encoded prior to the crossover.

5 A COMPLETE CDC SOLUTION

Current methodologies that focus on timing closure run the risk of re-spins and iteration due to the clock domain issues previously discussed. However, timing closure ensures that the violations within a clock domain are fixed, while those between the domains are false paths, that is, unchecked. As part of the verification strategy, the synchronization errors — i.e., between clock domains — should be eliminated at the RTL stage itself. In the timing-closure–only methodology (see *Figure 12*), these errors are checked during the end stages of the design cycle, i.e., in the gate simulation with SDF backannotation, a static verification tool adds value.

A complete CDC solution addresses all the functional aspects of multiclock SoC design verification. The structural checks are done for *s*CDC issues, and formal analysis is used to validate the *f*CDC issues.

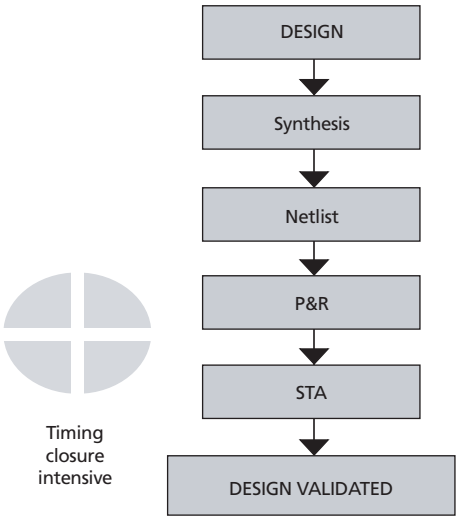

Figure 12: Timing-closure–only methodology

Adding CDC verification in the early design stages verifies and validates the unverified portion of the design (see *Figure 13*).

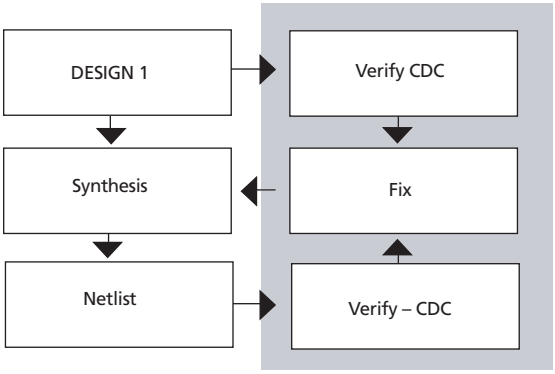

Figure 13: Methodology with CDC verification

5.1 CLOCK DOMAIN PARTITION

The major step in the setup for a CDC check is the proper partitioning of designs into asynchronous domains. Propagation and extraction techniques can aid the user in partitioning. Propagation is the forward flow of the user-defined clock attributes through different design structures. The user can control the forward propagation. Extraction builds the domain partition by starting from the clock pin of each flop and creating different clock trees.

Because the possibility exists that more domains could be created than the designer envisioned, a utility should be provided to associate clocks and data pins and ports. The clock domain partitions also depend on constraints in the design for the correct flow of the paths, so there should be a way to constrain the design and declare static signals. Configuration register information should be used in conjunction with the utilities to correctly define the clock domain partitions.

5.2 CDC PATH RULES AND VALIDATION

Once the clock domains are properly identified, the CDC paths become apparent. A rule-based technique can then be used and applied to the CDC paths. The rules can specify the synchronization scheme (flop or MUX), the allowable structures in the crossover path and the paths in the synchronizer, and the area of application of the rule. Tools can provide either global or local rules.

A CDC path is extracted and the set of user-specified rules is applied. It is either a flop synchronizer, MUX-based synchronization, or user-defined synchronizer module; if any rule passes, this CDC path is validated. The structures in the crossover path and the metastable path are analyzed based on the user-specified rule. The reconvergence of CDC signals should be applied only to the flop-based synchronizers, and analysis should start after the specified flop rule. A rule R1 may specify a three-flop synchronizer from CLK 1 to CLK 2; and another rule (for example, R2) may specify a two-flop synchronizer from CLK 3 to CLK 2. The reconvergence check should start analysis after three flops to data paths from CLK 1 to CLK 2 and two flops to data paths from CLK 3 to CLK 2, and see if the two CDC paths converge.

Once the structural analysis has been completed, the formal analysis and techniques can be used to verify the stability of all the signals in the CDC path. This formal analysis checks that the hold logic and the latching logic have correct functionality. For vectors that are flop-synchronized, formal analysis can be used to create a property to ensure that the vector is Gray-encoded.

5.3 FUNCTIONAL CHECK AND DETAILS

As shown in *Figure 14* and *Figure 15*, the functional check can be broadly classified into five checks:

- Source data (SD) stability
- Destination data (DD) stability

• MUX enable (ME) stability that applies only to MUX schemes

• Single-bit (SB) checks that apply only to vectors that are flop-synchronized

• Handshake (HS) assertion for the flop-based signals

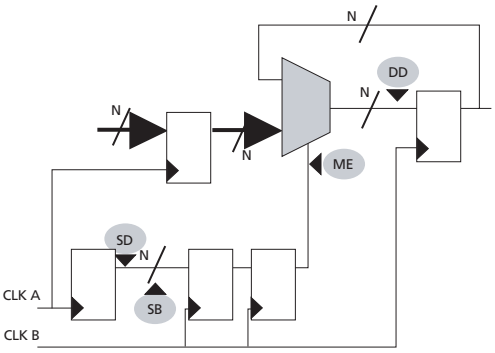

Figure 14: fCDC checks and their locations

5.3.1 Source data stability

The SD check ensures that the holding logic is functioning correctly; that is, the data is held properly until it is latched by the destination.

5.3.2 Destination data stability

If there is logic in the CDC path, even though the hold logic of each source is stable, the combination and computation could affect the holding at the destination. The DD check ensures that data at the destination is stable until it is latched.

5.3.3 MUX enable stability

For a MUX-based synchronization scheme, when the enable is activated, the data points in the MUX should not change. Hence, the output of the MUX should be stable until the destination domain captures the data.

5.3.4 Single-bit check

When a vector crosses over into an asynchronous domain, the vector is usually Gray-encoded. The single-bit check ensures that this rule is applied to all control vectors, and formally proves that there is one bit change under any given scenario.

5.3.5 Handshake check

The HS check (see *Figure 15*) looks at two or more CDC paths and the data flow, and checks to see that there is a response to all the transmitted signals. That is, it checks for a transmit-receive protocol. This check involves intense user intervention, because automatic analysis of signals involved in a handshake protocol is not trivial.

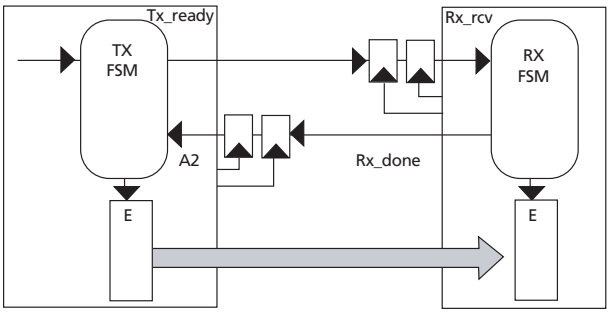

Figure 15: Handshake check

5.4 DIAGNOSIS

Failures should be viewed with a proper schematic viewer that uses color-coding to depict different domains for ease of use. Also, backannotation to the source code helps in debugging and fixing the code. The waveform can be shown for the failures in the functional stability checks.

6 EDA TOOLS

Many EDA tools with different business models are on the market. EDA vendors include Cadence, 0-In, Atrenta, @HDL, Mentor, and Synopsys. Cadence Encounter Conformal CDC capability (part of Encounter Conformal ASIC Equivalence Checker) is one of the leading products in this area. Its strength lies in its quality, flow, ease of diagnosis, and gate-level modeling expertise drawn from Encounter Conformal LEC (Logic Equivalence Checker). Leading SoC design houses are using the product to create complete CDC solutions.

6.1 ENCOUNTER CONFORMAL CDC CAPABILITIES

The Encounter Conformal CDC verification solution performs the following functions:

- Clock domain partition and topology checks
  - Proper clock tree definition and propagation
  - Known and unknown generated domains
- Structural checks for CDC path validation
  - Proper implementation of synchronizers to prevent metastability problems
  - Checks for convergence in the crossover path
  - Checks for divergence in the crossover
  - Checks for divergence of metastable signals
  - Checks for reconvergence of synchronized signals
- Functional checks
  - Proper data stability across clock domain boundaries, for both source data stability and destination data stability
  - Proper MUX enable stability across clock domain boundaries
  - Single-bit change (Gray encoding) checks for vectors
- Extensive diagnosis capabilities

6.2 EXAMPLE

Figure 16 shows how the Encounter Conformal CDC solution detects the structural design issues with synchronization (sCDC), as previously discussed. Encounter Conformal CDC detects a convergence in the crossover to-path (as explained in section 3.3).

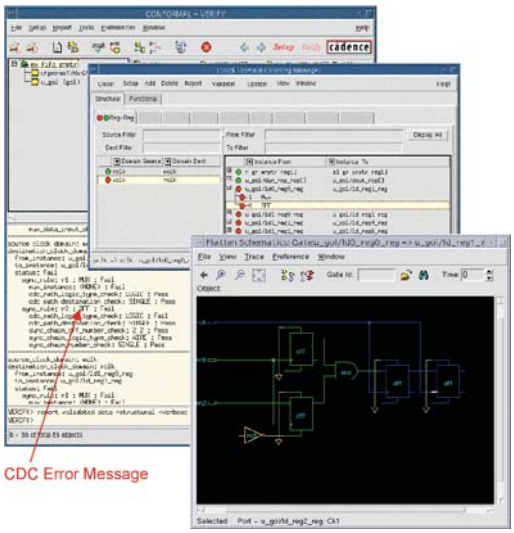

Figure 16: Encounter Conformal CDC detects structural design issues with synchronization

7 CONCLUSION

With a growing number of clocks in today's SoC designs, increased design complexity, and pressure for first silicon success, all clock and timing issues have become a verification challenge.

Existing methods provide an ad hoc partial verification that is time consuming and error prone. Automatic formal verification techniques are needed to ensure that multiclock designs are correct prior to tapeout. The solution must also prevent sources of failures in multiclock designs, such as metastability.

A clock domain crossing (CDC) verification solution must address this verification challenge, while maximizing overall productivity and effectiveness. It needs to cover clock domain analysis and structural and functional verification, addressing both RTL and gate-level verification needs. With the Cadence Encounter Conformal CDC solution, development teams can reduce the overall verification effort and the risk of related re-spins.

The Encounter Conformal CDC solution includes the following benefits:

- Pinpoints problems quickly
  - Automatic detection of clock domains and crossings
  - Structural verification of multiple clock domain synchronization
  - Functional verification for data stability violations
- Automates error-prone manual post-static timing analysis process
- Reduces risk of clock-related re-spins
- Prevents late clock-related iterations in the design cycle

8 REFERENCES

Metastability in Altera Devices. Altera Application Note 42. May 1999.

Bahukhandi, Ashirwad. *Metastability.* Lecture Notes for Advanced Logic Design and Switching Theory. January 2002.

Cummings, Clifford E. *Synthesis and Scripting Techniques for Designing Multi-Asynchronous Clock Designs.* SNUG 2001.

Haseloff, Eilhard. *Metastable Response in 5-V Logic Circuits.* Texas Instruments Report. February 1997.

Nystrom, Mika, and Alain J. Martin. *Crossing the Synchronous-Asynchronous Divide.* WCED 2002.

Patil, Girish, IFV Division, Cadence Design Systems. *Clock Synchronization Issues and Static Verification Techniques.* Cadence Technical Conference 2004.

Smith, Michael John Sebastian. *Application-Specific Integrated Circuits.* Addison Wesley Longman, 1997, Chapter 6.4.1.

Stein, Mike. *Crossing the abyss: asynchronous signals in a synchronous world.* EDN design feature. July 24, 2003.

Wakerly, John. *Digital Design Principles and Practices.* Prentice Hall, 2000.

Cadence Design Systems, Inc.

Corporate Headquarters
2655 Seely Avenue
San Jose, CA 95134
800.746.6223
408.943.1234
www.cadence.com

5472B 12/04