Full Flow Clock Domain Crossing - From Source to Si

M. Litterick
Verilab GmbH., Munich, Germany

***Abstract*- Functional verification of clock domain crossing (CDC) signals is normally concluded on a register-transfer level (RTL) representation of the design. However, physical design implementation during the back-end pre-silicon stages of the flow, which turns the RTL into an optimized gate-level representation, can interfere with synchronizer operation or compromise the effectiveness of the synchronizers by eroding the mean time between failures (MTBF). This paper aims to enhance cross-discipline awareness by providing a comprehensive explanation of the problems that can arise in the physical implementation stages including a detailed analysis of timing intent for common synchronizer circuits.**

I. INTRODUCTION

Functional verification of clock domain crossing (CDC) signals is normally concluded on a register-transfer level (RTL) representation of the design. However, physical design implementation during the back-end pre-silicon stages of the flow, which turns the RTL into an optimized gate-level representation, can interfere with synchronizer operation or compromise the effectiveness of the synchronizers by eroding the mean time between failures (MTBF).

Effective and robust CDC implementation and verification therefore becomes a multi-discipline problem, spanning the front-end processes of concept, design and functional verification and the back-end processes of logic synthesis, place & route, physical synthesis and design-for-test (DFT). This paper aims to enhance cross-discipline awareness and cooperation by providing a comprehensive explanation of the problems that can arise in the physical implementation stages and by exploring the connection to the design intent that is implicit in the front-end expectations. In order to encapsulate the scope of the back-end concerns, the paper concludes with an overview of DFT concerns for manufacturing test of CDC signals in modern technology libraries.

II. PRE-SILICON FLOW

An overview of the pre-silicon design, verification and back-end implementation flow is shown in Fig. 1. The diagram is simplified compared to reality (where there are more feedback loops, trial implementations, and steps could be performed in different order), but fulfills the purpose of putting the full flow into context in order to introduce the terminology used throughout this paper.

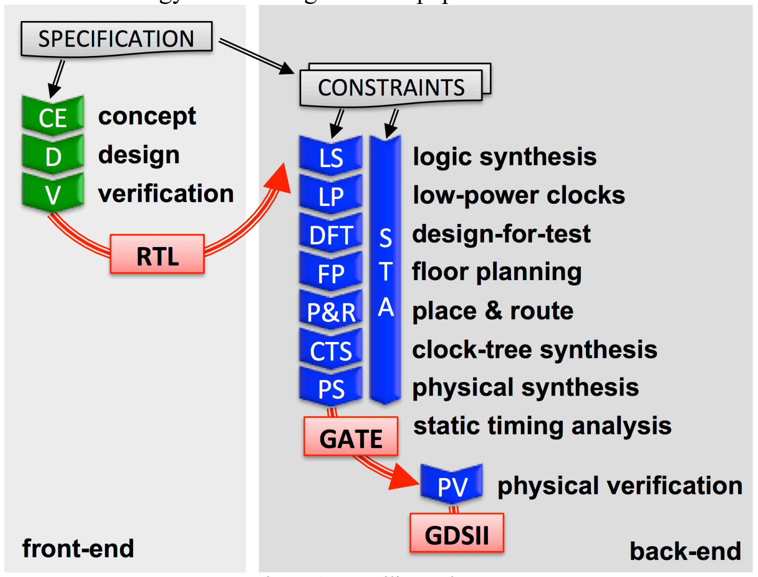

Figure 1. Pre-Silicon Flow

The key characteristics of each step are :

1) *Concept or architectural engineering is responsible for mapping the application to design requirements and includes tasks such as mathematical modeling, hardware/software partitioning, and power management and clock domain definition.*
2) *Design involves implementing the requirements in hardware description language code as an RTL representation of the design. This step involves digital design, analog schematic and model implementation, block-level design, synchronizer implementation, and integration of existing blocks into the top-level chip.*
3) *Functional verification is responsible for validating all aspects of functional operation of the RTL and analog blocks. Verification typically invokes formal and simulation techniques in order to configure, stimulate, check and cover the necessary conditions. CDC protocol, synchronizer operation, and operation in presence of synchronizers are validated at this stage.*
4) *Logic synthesis is the process of converting an RTL representation of the design into a netlist comprising logic gates, flip-flops, I/O buffers and interconnect. This step concludes with formal functional equivalence checking to validate that the two representations are logically identical. The mapping of synchronizer logic to technology cells is performed at this stage.*
5) *Low-power clock-gating involves automatic insertion of local clock gates to minimize the power consumption in register-based design portions where the content changes very infrequently.*
6) *DFT insertion is responsible for replacing flip-flops in the design with scan, or built-in self test (BIST) equivalents. This enables much more comprehensive manufacturing test. Additional logic is inserted to control clocks, mask resets, prevent floating signals, and avoid contention during scan operations. The scan chain order following scan insertion is fairly arbitrary, but is reordered at a later stage in the physical synthesis flow.*
7) *Floor planning involves coarse placement allocation based on connectivity rules with a view to minimize routing congestion and overall chip area.*
8) *Place & route performs timing driven placement of all the ports, cells and hard macros as well as connection and routing of the wires in the netlist.*
9) *Clock-tree synthesis is a process which converts the ideal clock nets used during RTL and initial place & route stages into clock-tree networks of buffers, inverters and interconnect capable of distributing the clocks throughout the design with acceptable skew characteristics. Clock-tree synthesis is also responsible for ensuring that synchronous derived clocks, which have different frequencies but aligned edge transitions, are balanced correctly since no synchronizer elements are used between these domains.*
10) *Physical synthesis is an additional synthesis stage used to optimize the gate-level netlist based on detailed placement and clock-tree information. This operation results in modified placement, cell substitution, and modified routing including scan chain reordering, in order to meet the timing, area, power and routing requirements. The output from physical synthesis is the final gate-level netlist and a graphical description of the layout (GDSII or similar).*
11) *Physical verification is executed to validate correctness of the graphical description for the layout. This step involves design rule checks (DRC), layout verses schematic checks (LVS), electrical rule checks (ERC), as well as some other physical checks such as oxide migration and density validation.*
12) *Static timing analysis is a mechanism for validating timing parameters in a gate-level representation of a chip. It is performed throughout the back-end flow using continually refined timing constraints.*

III. CDC Design & Functional Verification

This section provides a brief overview of CDC problems, synchronizer operation and functional verification in order to put the rest of the paper content into context. A full tutorial on the corresponding topics is outside the scope of this paper.

A. *Metastability & MTBF*

The fundamental issue for CDC operation is caused by metastability when a signal passing from one clock domain to the other violates the setup and hold times for the target flip-flop. This results in an unstable state on the flip-flop output which will eventually resolve to either a one or a zero after some delay, but the actual value cannot be predicted. Fig. 2 illustrates the effect of metastability:

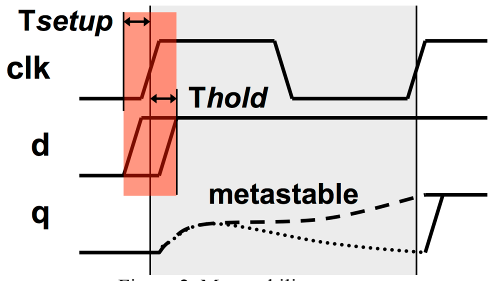

Figure 2. Metastability

Metastability is not observable in RTL or gate-level simulations, or during formal analysis, but is a real-life attribute of the flip-flop implementation. Metastability results in observable effects in the real application such as: uncertainty of value change time, filtering of pulses that were observed in simulation, generation of pulses that were filtered in simulation, and multiple timing scenarios resulting from CDC jitter observed on convergent synchronization paths [1].

The probability of a metastable event decaying to a valid (but unpredictable) value in time for the next sampling event is expressed by the mean time between failures (MTBF) equation shown in (1).

$$MTBF = \frac{e^{Ts/Tc}}{TwFcFd} \tag{1}$$

Where:

Ts = settling window

Tc = settling time constant

Tw = window of susceptibility

Fc = destination clock frequency

Fd = data change rate

It is important to note that the MTBF expression does not predict the rate of metastable events, but rather the probability of the metastable condition not decaying to a stable legal values in time for the next sampling event. A failure is therefore a catastrophic event, since unknown intermediate values will be sampled and propagated throughout the destination domain - but such a failure is also extremely infrequent compared to the relatively frequent metastable conditions.

B. Synchronizer Operation

Metastability cannot be avoided, it must be designed around. Synchronizers achieve reliable operation, with an MTBF of many years, by isolating the potentially metastable nets in such a way that the effects are not propagated throughout the destination clock domain. Typically CDC protocols require careful synchronization of individual control signals and careful management of associated data busses used to transfer information between the corresponding clock domains.

Optimal CDC strategy and synchronizer implementation for a particular application spans multiple disciplines including concept/architectural engineering, RTL design, and back-end implementation.

C. RTL Verification

Functional verification of CDC is normally performed on the RTL representation of a design; this is relatively early in the overall flow, the RTL is less complex than the gate-level representation, and most functional defects are observable at this stage. CDC verification involves structural analysis (to identify clock domains, CDC signals, and illegal logic on CDC paths), classification of CDC signals into synchronizer protocol groups (such as individual asynchronous bits, handshake synchronizers or asynchronous FIFOs), determination of appropriate properties for each protocol, and validation of these properties by simulation or formal analysis [1]. It is not enough to validate the correct operation of the synchronizers in isolation, we also have to ensure that the rest of the device works in the presence of the synchronizers - including all clock relationships, effects of uncertainty, and convergent paths through multiple synchronizers in the presence of CDC jitter [1].

D. *Physical Effects*

In fact functional verification of CDC at the RTL stage is necessary, but not sufficient. Back-end processes such as logic synthesis, place & route, physical synthesis, low-power clock gating and design-for-test insertion can interfere with synchronizer operation and CDC integrity. Even when the CDC analysis is concluded on a clean RTL implementation, the back-end physical processes can reduce the effectiveness of the synchronizer or stop it working entirely. The solution however is not gate-level simulation on a back-annotated netlist, but rather structural and timing analysis of the final implementation with a view to validating the attributes necessary for correct CDC operation.

This multi-discipline problem is exacerbated by the fact that RTL is designed to be generic (it will work with any reasonable clock frequency, given the right technology resources), whereas the physical implementation is product-specific (actual clock frequencies, given technology library, physical resource limitations in terms of area and power budgets). So even though some of the CDC intent should be communicated from RTL to physical flow, it might not be appropriate to encapsulate this information in the RTL domain itself.

IV. Static Timing Analysis

Static Timing Analysis (STA) is a mechanism for validating all legal timing parameters in a gate-level representation of a chip. STA performs a structural analysis of the netlist in order to identify all possible timing arcs, and uses a timing constraints file as a reference description of the required timing. The timing constraints file is initially manually generated to match the device specification and provides a project-specific target implementation goal for the generic RTL representation of the design. During the evolution of the netlist (from logic synthesis through place and route, clock-tree synthesis to physically synthesis) the timing constraints are continually enhanced to provide more accurate representation of the requirements for the device.

The main timing arcs considered by STA include timing between storage elements in synchronous designs (i.e. flip-flops or latches) and timing between primary inputs and outputs relative to these internal registers, however absolute path and sub-cycle timing can also be validated. Fig. 3 illustrates the primary timing arcs for communication between two flip-flops, where *Tck_skew* is the clock skew after clock-tree synthesis, *Tck_q* is the clock to output delay for the launching flip-flop, and *Tpath* is the total path delay including combinational gates and wire delays.

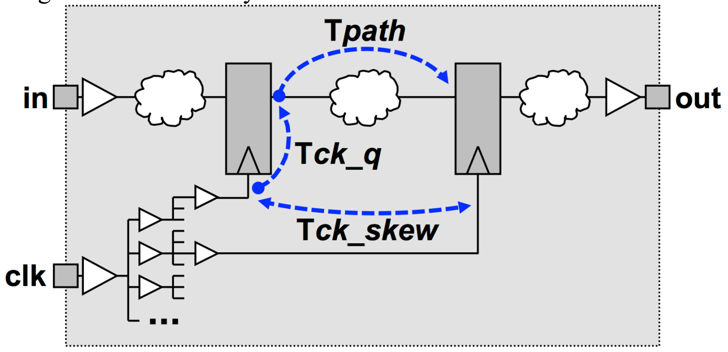

Figure 3. Timing Arcs

Note that the timing is not just analyzed against a single set of parameters, but the tools and implementation must consider multiple possible conditions including process variations during manufacture, minimum and maximum operating voltage and temperature, as well as on-chip variations due to the manufacturing process and operating conditions (such as temperature gradient across the die). Under best-case timing conditions hold-time analysis is critical, under worst-case conditions setup-time becomes relevant.

Most complex designs do not have a single set of timing constraints, but rather they implement a multi-condition multi-mode (MCMM) strategy, whereby several sets of timing constraints exist for different operating modes of the device (e.g. normal functional mode and manufacturing test mode). This means that not all of the timing constraints are present all of the time, and it allows analysis of timing arcs which are only sensitized in one mode to be ignored under other conditions.

Note that if signal paths are marked as "false paths" then the physical implementation will treat them as low-priority unconstrained nets and STA will ignore them during analysis, hence they are effectively treated as “don’t care” conditions and this may result in bad placement, poor routing and weak drivers (which may of course be acceptable, if you don’t care).

A. STA Within A Single Clock Domain

Clock-tree synthesis ensures that all the clock signals within a single clock domain occur with acceptable skew to allow for synchronous operation. The primary concern for STA within a single clock domain is ensuring that the physical implementation meets both the hold time for the current clock edge and the setup time for the next clock edge in order to guarantee synchronous communication between the flip-flops. However, for CDC synchronizer circuits (such as the 2-flip-flop synchronizer shown in Fig. 5), it is not sufficient to meet the normal synchronous timing requirements for the potentially metastable net even though these flip-flops are actually within the same clock domain. If the timing on the metastable net path is close to the normal synchronous requirements, then the resulting settling window (*Ts*) is small which dramatically reduces the MTBF since *Ts* has an exponential effect on MTBF as shown in (1). The settling window can expressed as (2).

$$Ts = Tperiod - Tck\_skew - Tsetup - Tck\_q - Tpath \tag{2}$$

Additional sub-cycle timing constraints are required to ensure that the maximum delay for the metastable net under normal conditions is much less than the clock period, in order to create an adequate settling window opportunity as shown in Fig. 4.

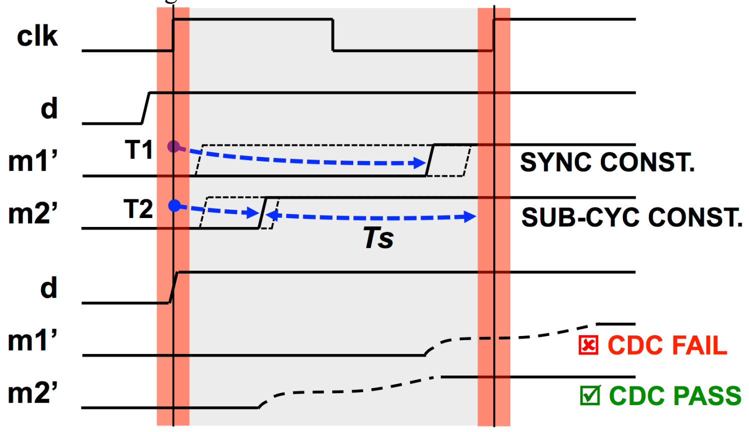

Figure 4. Sub-Cycle Timing For Metastable Net

Fig. 4 shows an example of normal synchronous timing (the dotted box around *T1*), sub-cycle timing (the box around *T2*) for a potentially metastable net (*m*), and the need for settling window (*Ts*) when a metastable event occurs. In the lower half of Fig.4, the *d* input violates the setup and hold timing for the flip-flop resulting in metastability, if the metastable net is constrained only to meet synchronous timing (*m1*’), then the net delay (which, for simplicity, is shown as a transport delay in the diagram) can result in an insufficient settling window, and the metastable state is propagated beyond the next sampling event - this represents a failure of the CDC circuit. If however the metastable net is constrained with a sub-cycle path constraint (*m2*’), then the net delay and metastable decay time are still concluded within the settling window, representing a normal CDC pass condition. Such an additional sub-cycle timing constraint is a good example of design intent that needs to be communicated between design and back-end implementation. Physical synthesis and other back-end processes need to honor this additional sub-cycle timing requirement, especially since the normal synchronous timing requirements would be easy to meet using weak drivers and low-priority routing for the metastable net (which has no combinational logic and a fan-out of one), which would destroy synchronizer effectiveness.

B. STA Between Multiple Clock Domains

The primary observation for asynchronous clock domains is that there is no fixed timing relationship between the clock signals and therefore:

- *clock skew between the domains is not fixed, but changes continually*
- *setup and hold times will frequently be violated at the boundaries between domains*

Synchronizers are designed to cope with the effects of metastability when the setup and hold times are violated as a result of the incidental clock relationship between the two clock domains when changes on the

CDC signals occur. Because there is no timing relationship between the clocks, there is no inter-domain path delay timing requirement for CDC signals passing between the two clock domains [2], but that does not mean that all timing characteristics for the CDC signals are irrelevant. Specifically, we care about:

- *transition time of the CDC signal at the destination flip-flop*
- *timing relationships between signals in CDC protocol groups*

The transition time for the CDC signals is important because slow rise and fall times on the data inputs to a flip-flop increase the setup and hold requirements. This in turn increases the effective window of susceptibility (3), which degrades the synchronizer MTBF (1).

$$Tw = Tsetup + Thold \tag{3}$$

If no timing requirements are specified for the CDC signals (i.e. they are marked as false paths), or a very slack timing constraint is provided (e.g. for slow-speed scan test), then the physical synthesis can implement very weak drivers and low-priority routing for the CDC net which can result in very slow transition times.

Likewise, if no timing relationships are specified in the constraints for all of the signals comprising a complex CDC protocol (i.e. synchronizers with control and data paths), then some or all of these signals could incur degraded timing due to physical synthesis process, resulting in incorrect synchronizer behavior. Some examples of this are provided in the next section.

V. Physical Requirements of Synchronizers

This section provides an analysis of physical requirements for a few of the most common synchronizer designs with a view to exposing potential hazards and pitfalls that can occur during back-end implementation. Additional STA constraints and checks are required to ensure valid operation of the corresponding circuits in real-world applications.

A. Two Flip-Flop Synchronizer

The 2-flip-flop synchronizer (2FF) shown in Fig. 5 is the fundamental synchronizer element for individual asynchronous signals and a key building block for many more complex protocol synchronizers.

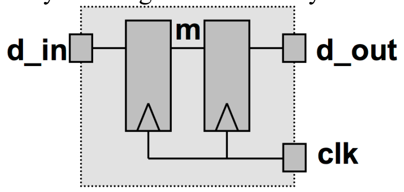

Figure 5. 2-Flip-Flop Synchronizer

The main functional requirements for the 2FF synchronizer which can be validated using formal or simulation techniques with the RTL representation of the design include [1]:

- *no combinational logic on CDC path that could be falsely sampled*
- *no narrow pulses on CDC signal that are never sampled by synchronizer*
- *data stability must be greater than 2 active clock edges to ensure no filtering effects*

Additional physical requirements for optimal operation of the 2FF synchronizer, roughly in order or importance, are:

- *sub-cycle timing required for metastable net (between 1st and 2nd FF)*
- *select metastability hardened implementation for 1st FF*
- *select monolithic macro-cell for complete 2FF synchronizer*
- *fast transition time is required on CDC signal at input to 1st FF*
- *do not scan insert the 2nd FF*
- *clock-tree synthesis has tight skew requirements between both FFs*

The metastable net in the synchronizer should achieve much stricter timing than that required by the normal synchronous operation for the destination clock domain otherwise the settling time window can be crucially eroded resulting in compromised MTBF (as discussed in Section IV A). There are implicit additional mapping, placement and routing requirements on the metastable net and the two FFs in the synchronizer, but the end result is really ensuring the sub-cycle timing is appropriate. Selecting an appropriate value for the sub-cycle timing constraint of the metastable net is not an exact science and depends on the acceptable MTBF for the application. Several possibilities exist, including:

- *maximum delay based on technology library capability (e.g. minimum reasonable)*
- *maximum delay as a portion of the minimum clock period (e.g. 20%)*
- *absolute value based on MTBF calculation (i.e. if Tc and MTBF requirements are known)*

Where possible the synthesis should be directed to select, and stick with, a metastability hardened flip-flop for the 1st FF in the synchronizer. These metastability hardened flip-flops typically have a faster settling time constant (e.g. by using an overdriven feedback loop) and narrow windows of susceptibility (i.e. small setup and hold times). The settling time constant (*Tc*), which affects the damping of metastable conditions, has an exponential effect on the MTBF, whereas the narrow window of susceptibility (*Tw*) has a linear effect (1). Note that it might not be possible to translate some RTL logic constructs to the metastability hardened flip-flops that are available in the target library (e.g. there may be no such flip-flops with reset capability, or scan equivalents, since both of these features affect MTBF), so this needs to be considered at design time.

Some technology libraries provide pre-packaged 2FF synchronizer elements as monolithic hard macro-cells. Where possible synthesis should select these macro-cells as direct replacements for the corresponding RTL constructs and preserve them in the netlist. The macro-cells are likely to instantiate a metastability hardened flip-flop for the first stage, and ensure the sub-cycle timing requirements are well encapsulated and protected from interference. The mapping to such 2FF macro-cells may also need to be considered at design time since not all possible combinations (e.g. reset or scan) will be available in the library.

Some design flows instantiate metastability hardened flip-flops or macro-cells for the 2FF synchronizer directly into the RTL; this provides encapsulation of most of the timing and physical concerns and it simplifies the synthesis process, but compromises the portability and flexibility of the RTL by making it technology-dependent. Typically this limitation is manageable for internal designs and derivatives, but the approach could be inappropriate or even impossible when using or supplying third party IP blocks.

As discussed in Section IV B, very slow transition times on the CDC signal when it arrives at the synchronizer can compromise MTBF because the window of susceptibility (*Tw*) is increased as a result. Hence there is an implicit requirement to either insert buffers on the CDC signal to smarten up the edge rate (remember the path delay is not a fundamental timing requirement, so buffer insertion would have no drawbacks), or ensure cell mapping for the driver, placement and routing of CDC signal result in appropriate transition times at the 2FF input. Remember the critical thing here is that for normal STA the CDC signal represents a false path, but if marked as such then the synthesis tools are allowed to reduce drive strength and relax routing requirements. Typically the only actual timing constraint is that required for stuck-at fault testing using scan mode and MCMM analysis, which would not be enough to maximize synchronizer performance.

Scan insertion is another potential source of MTBF degradation, since the scan equivalent flip-flops have additional multiplexer logic on the data path and additional scan routing could increase the load on the driver. In fact scan insertion of the 1st FF has no detrimental effect, it is only scan insertion of the 2nd FF which erodes the settling window and affects the MTBF. However scan insertion is redundant in both cases since the source and destination flip-flops are connected in a shift register formation with no intervening logic and therefore can be directly controlled without additional scan paths when absorbed into a full-scan implementation, i.e. there is no need for a scan-enable control when there is no combinational logic to bypass.

There are contradicting requirements on the clock skew for best and worst case timing which means that clock-tree synthesis has to ensure very tight timing for the clocks in the 2FF synchronizer. Specifically, under best case conditions there is a danger of hold-time violation at the 2nd FF since there is no logic on the metastable path, a hazard that is made worse if the clock to the 2nd FF occurs after the clock to the 1st FF. Likewise, under worst case conditions any skew that results in the clock to the 1st FF coming after the clock to the 2nd FF will erode the available settling window (*Ts*) resulting in reduced MTBF (in this case the absolute skew value is expected to be small, but the effect is exponential). The best results would be obtained with the maximum positive skew (2nd FF clocked after 1st FF) that just meets the hold-time requirement, but more pragmatically by minimizing the skew tolerance between these two flip-flops.

B. Parallel Synchronizer

The parallel synchronizer shown in Fig. 6 can be used in applications where the multi-bit input data is gray-coded, and it is the key synchronizing element in Asynchronous FIFO Synchronizers [1][2]. The parallel synchronizer will not generate any illegal codes (i.e. bus values) in the destination domain due to independent metastability uncertainty in each of the 2FF blocks provided only one data bit changes at a time (hence only one bit can go metastable and resolve to either the old or the new value). This synchronizer cannot be used to synchronize binary bus values since illegal values can be generated on the destination side if two or more bits change close to a sampling edge.

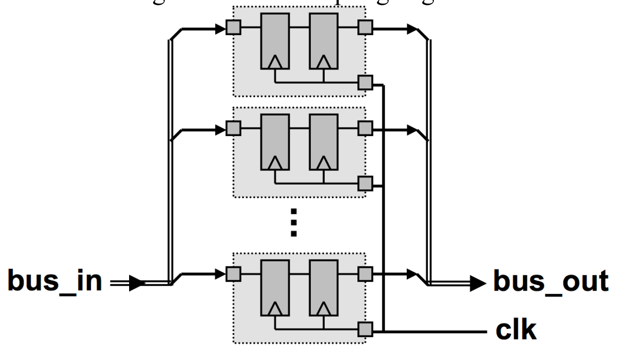

Figure 6. Parallel Synchronizer

The main functional requirements that are validated at the RTL level are:

- *no combinational logic on CDC path that could be falsely sampled*
- *the data must be gray-coded in the source domain*

Note that although the data codes are all valid and in the correct order inside the destination domain, some codes may be skipped if the source clock is faster than the destination clock, hence gray-coding checks are not appropriate inside the destination domain and the glitch and data stability checks from the standard 2FF synchronizer are also not appropriate.

Each 2FF block inherits the physical requirements discussed in the previous section, and there are additional physical requirements for the parallel synchronizer:

- *relative data path variance for all CDC bits must preserve gray-coding at destination*

The primary concern for the physical implementation of the parallel synchronizer is that if the individual bits in the CDC bus are under-constrained, then the physical optimization can introduce significant skew into the relative path delays for each bit on the bus resulting in a functional failure for the synchronizer. Many constraint problems could create this scenario, including marking the CDC bus signals as false path (i.e. unconstrained), defining them as multi-cycle paths, using the clock period from a slower destination domain, or reverting to large default timing constraints (e.g. for slow-speed scan mode). For example, if the constraint range is too large and the actual data path is relatively slow for one bit on the bus, but relatively fast for another bit on the bus, then we can end up with signal reordering or multiple effective bit changes seen by the destination domain as shown in Fig. 7, which results in a functional failure of the synchronizer.

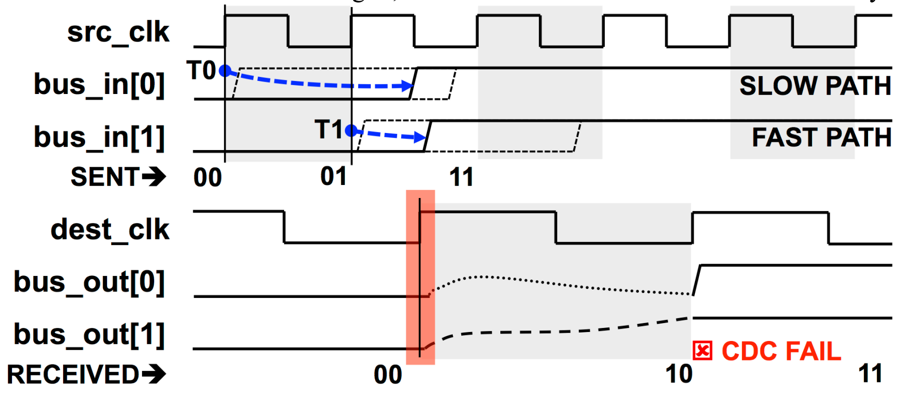

Figure 7. Parallel Synchronizer Failure Due To Path Variance

Note that there is no absolute path delay requirement from the source to destination for any of the individual bits in the parallel synchronizer, but rather a relative timing requirement that no two signal changes happen within the window of susceptibility of the destination clock domain as described by (4). (Where the subscript 1 refers to the source domain, 2 refers to the destination flip-flop, and B represents the later changing signal than A.)

$$[(Tperiod_1 + Tck\_q_{1B} + Tpath_{1B}) - (Tck\_skew_1 + Tck\_q_{1A} + Tpath_{1A})] > (Tsetup_2 + Thold_2) \quad (4)$$

In practice it is difficult to enforce a relative data skew constraint without a path constraint, and therefore it is easier to over-constrain the path delays on the CDC signals for the parallel synchronizer configuration. Clearly a multi-cycle path is inappropriate, therefore we need to provide a sub-cycle path constraint relative to the fastest source clock which takes into account the window of susceptibility on the destination side as shown in (5).

$$(Tck\_skew_1 + Tck\_q_1 + Tpath_1) < (Tperiod\_min_1 - Tsetup_2 - Thold_2) \qquad (5)$$

C. Handshake Synchronizers

Handshake synchronizers transfer multi-bit data values between the clock domains by synchronizing control signals and ensuring that the data bus is stable when the destination latches the data. Many different protocols are possible, but they all share the same characteristics that the actual CDC data does not pass through a 2FF synchronizer, but rather the associated handshake control signals do as shown in Fig. 8.

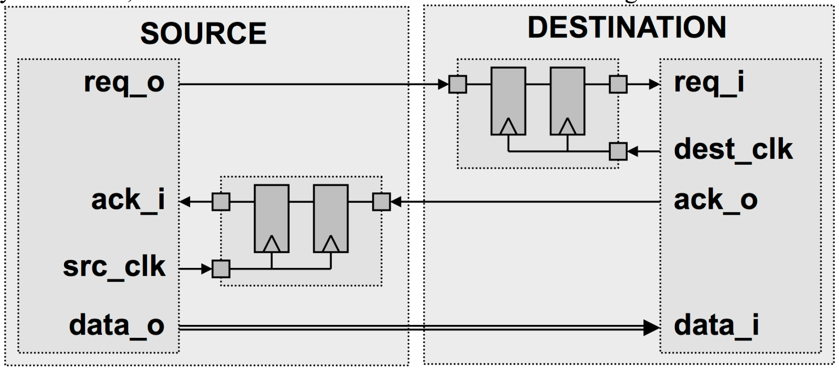

Figure 8. Handshake Synchronizer

The functional requirements that are validated at RTL level depend on the type of handshake synchronizer (full or pulse-based) [1], but include such concerns as:

- *request-acknowledge protocol (each req gets ack, no ack without req)*
- *full handshake protocol (req-ack-!req-!ack sequence for each data)*
- *data must be stable until acknowledge is seen in source domain*

In addition to the physical requirements for the 2FF synchronizers on the control signals, there are additional timing requirements for the CDC data signals relative to the control signal path. Specifically these signals must have delays that ensure the correct value is latched into the destination domain. If the CDC signals are marked as false path, have incorrect constraints relative to a slow source domain, or bad default constraints, then the physical implementation can result in long delays due to weak driver and poor routing which means the data could be latched incorrectly as shown for the upper *data* path in Fig. 9.

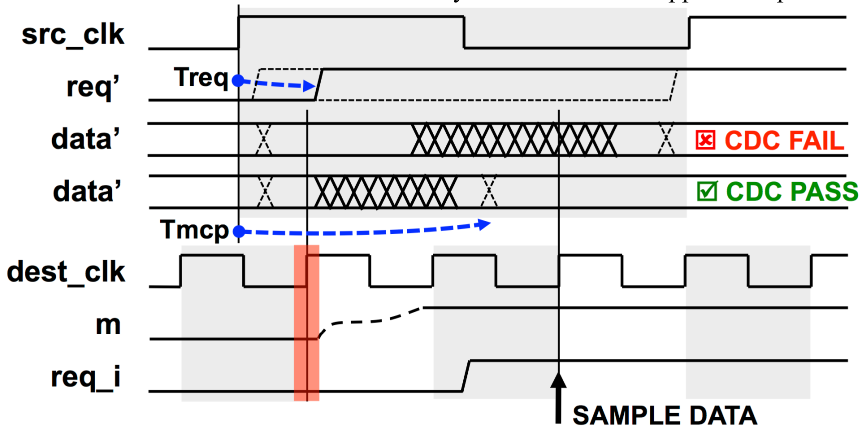

Figure 9. Handshake Synchronizer Failure Due To Slow Data Path

Notice in Fig.9 that the request (*req*) and failing *data* signal are constrained to one relatively slow source clock period (shown by the dotted boxes around the signals). In this case the *req* signal has a short actual delay (*Treq*) after physical synthesis and the last data bit a relatively long delay. In this example the *req* happens to assert just after a clock in the destination domain, but violates the hold-time, causing metastability which results in the earliest possible latching of the data (just less than two destination clocks after the *req*). So the data is sampled while the upper *data* is changing and the synchronizer protocol is broken. The lower *data* timing is constrained by a multi-cycle path constraint (*Tmcp*) relative to the destination clock frequency and is correctly sampled.

Since the RTL verification validates the data stability (i.e. the data must be stable until the acknowledge is observed in the source domain), then the physical requirements come down to a minimum path delay for each of the data signals. Specifically the delay on each CDC data signal must be shorter than the minimum time it takes for the far end to recognize the CDC request and latch the data, which is given by (6)

$$(Tck\_skew_1 + Tck\_q_{1D} + Tpath_{1D}) < [Tck\_q_{1REQ} + Tpath_{1REQ} + (2 * Tperiod\_min_2) - Thold_2] \qquad (6)$$

In this case the practical implementation is easier to achieve by accepting some minimal over-constraining. Specifically if the control signal path delay is greater than the hold-time for the destination flip-flop (which is very likely) then this delay can be ignored, and the data signals can be constrained by a multi-cycle path of two *destination* clock periods.

D. Phase Detect Synchronizers

Phase detect synchronizers such as those shown in Fig. 10 are used for mesochronous and plesiochronous applications where the source and destination clock frequencies are nearly the same [5]. A full analysis of the functional operation of these synchronizers is outside the scope of this paper, but they are included here since they demonstrate another example of sub-cycle critical timing for physical implementation.

These synchronizers all rely on similar mechanism - first detect the phase separation of the clocks, if they are close such that a timing conflict can arise, then shift either the data or the clock to avoid metastable events. If the phase offset does not vary over time (mesochronous case) then the phase detection and compensation only needs to be performed once. If the phase offset varies over time, due to a slight difference in frequency, (plesiochronous case) then the phase detection and compensation is a continuous ongoing process.

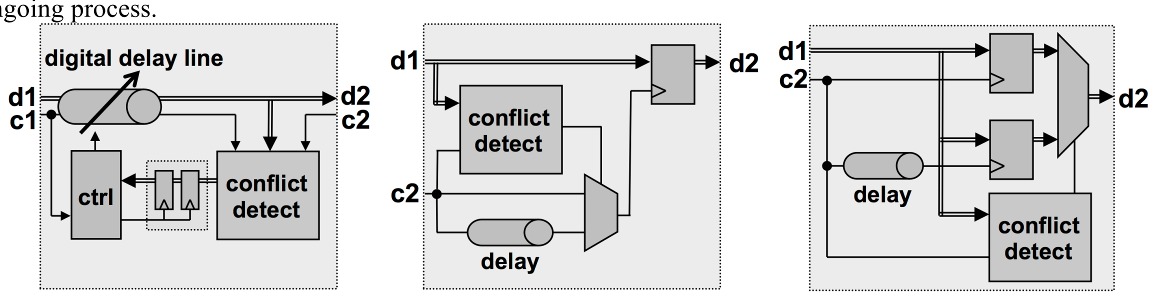

Figure 10. Phase Detect Synchronizers

In addition to implementation-specific physical requirements for the clock conflict detection macro-cells, each of these synchronizers has an additional physical requirement to enable phase compensation:

- *sub-cycle timing is required on the CDC data signals*

Specifically, under worst case timing conditions, the CDC data signals must be constrained to less than one clock period in order to create a window of opportunity for adjusting the clock or data delays. So the CDC signals have tighter, sub-cycle timing than the general logic in each of the source and destination domains (which only has to meet normal synchronous timing requirements, with the same clock period in both cases). The absolute value of the sub-cycle timing constraint depends on the granularity of the phase compensation mechanism, for example if an N-stage digital delay line is used, then the maximum delay adjustment must be subtracted from the minimum clock period to determine the sub-cycle constraint.

VI. Design-For-Test & Scan Implementation

A full analysis of DFT techniques related to scan testing of CDC circuits and synchronizers is outside the scope of this paper, however it is necessary to provide an overview of this step in the back-end pre-silicon flow in order to close the loop on the back-end processes.

Two important observations for manufacturing test need to be considered. Firstly, manufacturing test does not typically test the actual operation of the synchronizers, metastability behavior or measure whether or not the target device has acceptable MTBF. This is because correct resolution of metastable events is unpredictable, even in a working synchronizer. Therefore the test process validates other aspects of synchronizer implementation such as transistor switching, timing and connectivity in order to ensure design intent is not only correctly implemented, but working in the actual device under test (DUT). MTBF may be evaluated (or at least deemed acceptable) in post-silicon characterization, but this is not used as part of the manufacturing test criteria for every device. Secondly, it is not sufficient to test only for stuck-at faults using slow-speed scan test methodology in modern technologies with small geometries. Specifically it is considered necessary to test for transition and path delay faults within each clock domain and on the CDC signals between the domains by utilizing at-speed scan techniques [4].

Many different schemes are possible for at-speed scan testing [3], including launch-on-shift and launch-on-capture techniques with one-hot, aligned and staggered variants. Each approach has advantages and disadvantages, but they all have one thing in common: during the scan shift phase the slow-speed clocks can be provided directly from the tester and an on-chip clock generator is responsible for generating the carefully aligned capture clocks. One such advanced scheme which caters for at-speed testing of intra- and inter-clock domain faults for asynchronous domains is called staggered launch-on-shift (or staggered double capture), and is shown in Fig. 11.

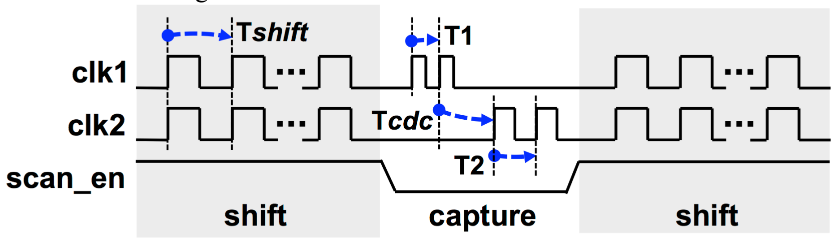

Figure 11. Staggered Launch-On-Capture At-Speed Scan

In Fig. 11, the shift clocks are provided directly from the test equipment, they are shown as aligned (but this is only necessary if the scan chains are stitched together) and run at a slow speed (*Tshift*). The two capture pulses on *clk1* allow for at-speed testing of the internal paths in the *clk1* domain (*T1*), the two pulses on *clk2* allow for at-speed testing of the internal paths in *clk2* domain (*T2*), and the separation of *clk1* and *clk2* (i.e. the staggered relationship between the second *clk1* capture pulse and first *clk2* capture pulse) allow for at-speed testing of CDC signal paths between the two domains (*Tcdc*). According to [3] the staggered double-capture approach is the preferred scheme for true at-speed testing of designs having a number of asynchronous clock domains, since it allows for manufacturing test of most transition and path delay attributes for CDC and synchronizer circuits.

VII. ACHIEVING CONTINUITY & CLOSURE

While the primary aim of this paper was to raise cross-discipline awareness on the potential back-end threats to synchronizer effectiveness, it is clear that additional methodology and techniques must be applied in order to achieve continuity and closure. One way of closing the gap between the disciplines is to ensure requirements as well as intent flow across the boundaries. Possible solutions include:

- *assertion-based functional timing checks running on RTL and gate-level*
- *CDC structural analysis executed on RTL and gate-level*
- *additional requirements for STA at all stages of back-end implementation*
- *at-speed test of CDC and sub-cycle synchronizer timing requirements*
- *explicit multi-discipline CDC and synchronizer sign-off review*

Concurrent assertions, written in a language such as *SystemVerilog Assertions (SVA)*, can be used to specify high-level timing requirements for functional operation of the synchronizer protocols [6][7], (note that this is a different usage model to normal SVA, which is event based). These assertions can effectively measure time domain properties such as pulse width, sub-cycle delays, and relative timing of signal changes and require access to project-specific implementation timing. Since the assertions can be bound to both RTL and final back-annotated gate-level netlist (although the actual binding may be different as discussed in [6]), they represent a good safety net for validating closure on some key implementation related artifacts using simulation techniques.

Some CDC analysis tools are now capable of operating on massive gate-level netlist representations of the design with manageable information output [8]. The main advantage of running CDC analysis on the final netlist (as well as the RTL signoff stage) is to pick up additional structural artifacts that were introduced during the back-end synthesis stages such as additional logic on CDC signal paths that might result in glitches, and badly constrained clock-tree synthesis which could for example destroy an intended derived clock structure. The tools use the same fundamental clock domain and synchronizer protocol descriptions as the RTL phase and can therefore assess if some high-level intent has been compromised.

All of the additional physical timing requirements highlighted in this paper (and others appropriate to the reader's application) need to be explicitly addressed by STA throughout the back-end process. This becomes a problem of constraints management, since the constraints themselves evolve throughout the implementation phases, and consecutive stages in the flow may use tools from alternative vendors requiring different constraint formats as discussed in [9].

Since at-speed testing of CDC signal paths and synchronizer elements is becoming a requirement for high-reliability CDC applications in the small geometries of modern technology libraries, there is an explicit need for manufacturing test to be aware of additional timing constraints required to achieve appropriate synchronizer performance. Typically however, the clock generation for at-speed operation needs to be done on-chip and therefore the information exchange is limited to the synthesis and DFT insertion teams.

Finally, we would recommend a methodology which involves an explicit dedicated CDC and synchronizer review as part of the final pre-silicon sign-off criteria prior to tape-out. This would allow all parties to bring their expectations of intent to the table, and ensure that not only does the device meet the criteria which were specified, but ensures that additional steps have been taken to maximize the performance and MTBF of the synchronizers in the actual application. Once this process is established, the cross-discipline awareness and general product quality for all current and future derivatives should benefit.

REFERENCES

[1] Litterick, "Pragmatic Simulation-Based Verification of Clock Domain Crossing Signals and Jitter Using SystemVerilog Assertions", DVCon 2006.

[2] Cummings, "Synthesis and Scripting Techniques for Designing Multi-Asynchronous Clock Designs", SNUG 2001.

[3] Wang, Wu, Wen, "VLSI Test Principles and Architectures", ISBN-10:0-12-370597-5.

[4] Lin et al, "High-Frequency, At-Speed Scan Testing", IEEE CS 2003.

[5] Semiat, Ginosar, "Timing Measurements of Synchronization Circuits", ASYNC 2003

[6] Litterick, "Using SystemVerilog Assertions in Gate-Level Verification Environments", DVCon 2006.

[7] Nordstrom, "Sub-cycle Functional Timing Verification Using SystemVerilog Assertions", SNUG 2013.

[8] Hughes, "Technology trends demand netlist-level CDC verification", Tech Design Forum 2015.

[9] Gangadharan, Dewangan, "Constraints Management: Approaches and Techniques for Preserving the Intent of Timing Constraints Throughout the Design Flow", Atrenta White Paper 2012