Static Analysis of Asynchronous Clock Domain Crossings

Shubhyant Chaturvedi
Advanced Micro Devices Inc.
Austin, Texas, USA
shubhyant.chaturvedi@amd.com

***Abstract*—Clock domain crossing (CDC) signals pose unique and challenging issues in complex designs with multiple asynchronous clocks running at frequencies as high as multiple giga hertz. Designers can no longer rely on *ad hoc* approaches to CDC analysis. This paper describes a methodical approach for static analysis of structural issues in asynchronous CDCs. The illustrated approach can be integrated easily in standard static timing analysis (STA) flows of any design house. The methodology was successfully deployed on a 32 nm accelerated processing unit (APU) design, and a case study of the same is included in this paper.**

Keywords - STA, CDC, clock domain crossing

I. INTRODUCTION

Data transfer across multiple asynchronous interfaces is a common scenario in system-on-a-chip (SoC) architectures, which integrate several IPs. With it comes the challenging task of clock domain crossing (CDC) verification, which can either be structural or functional [1][2]. Functional CDC verification relies on dynamic test vector-based RTL simulations and assertion-based verification, and is used to verify design issues related to re-convergence of synchronized signals, hand-shaking protocols, etc. [2][3]. Structural CDC verification involves tracing design topologies in a static way without the use of any test vectors. This paper illustrates a methodical approach for structural CDC verification that can be integrated easily in the static timing analysis (STA) flows of any design house without causing much churn.

The paper is divided into several sections. Section II gives an overview and background on key CDC issues. Section III describes the methodical approach in a step-by-step fashion to address some of the key structural CDC issues. Section IV describes our case study of this approach using an accelerated processing unit (APU) design, followed by a summary in Section V.

II. BACKGROUND

Clock domain crossings can occur between synchronous or asynchronous clocks. Synchronous clocks are clocks with the same, or an integral multiple of, frequency and either zero or constant phase-difference. Asynchronous clocks are clocks that do not have a definite frequency and phase relationship. Paths between synchronous clocks get timed in STA. The challenge lies in the verification of asynchronous CDC signals, which are left unconstrained and untimed in STA because these commonly get declared as false paths to the timing tool and have the potential to become metastable [1], as shown in Figure 1.

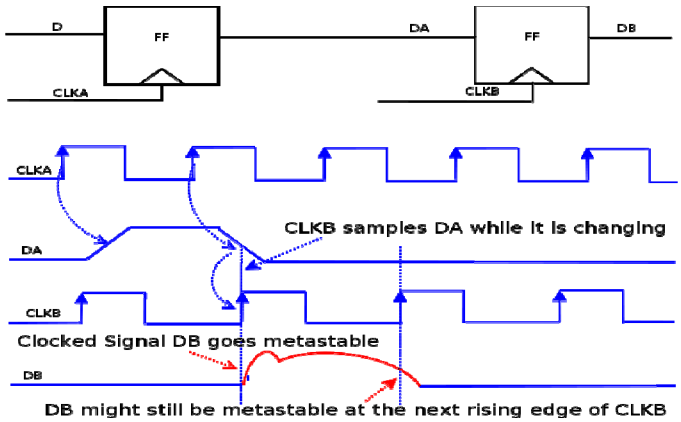

Figure 1. Metastability Issue

Metastability issues in single-bit asynchronous CDC paths are commonly resolved by deploying multi-flop structures known as synchronizers in the destination domain. On the other hand, for multi-bit CDC signals, more sophisticated synchronization schemes like Mux-D and FIFO are used [1][2][3], in which synchronized control signals are used to transfer data bits across clock domains, thus eliminating the need to deploy explicit multi-flop synchronizers on the multi-bit data signals.

The depth of the synchronizer - or, in other words, the number of latches needed in the synchronizer - can be determined by computing failures in time (FIT) for the cross-over path. The typical formula used for FIT computation [5][6][7] is:

$$FIT = \frac{10^9}{MTBF} = f_{clk} t_a r_{data} \exp\left[-\frac{(N_{latches} - 2)(t_{cycle}/2)}{\tau}\right] \quad (1)$$

Here, MTBF is the mean time between failures; f_{clk} is the frequency of the clock in destination domain; $N_{latches}$ is the number of latches used in the synchronizer; t_a is the aperture

978-3-9810801-8-6/DATE12/©2012 EDAA

time (the required time difference between data and clock to achieve a reasonable clock-to-output delay), and can be approximated as the sum of setup and hold time at the synchronizer input; r_{data} is the incoming data rate into the synchronizer; t_{cycle} is the cycle time of the destination clock; and, τ is the resolution time constant, which has a dependency on the operating and threshold voltages of the device. A detailed description of how to measure τ is given in [6], and we use a similar approach for our measurements.

Asynchronous CDC paths are untimed. Therefore, any combinational logic placed in these paths could be susceptible to glitches and data loss [1], as shown in Figure 2.

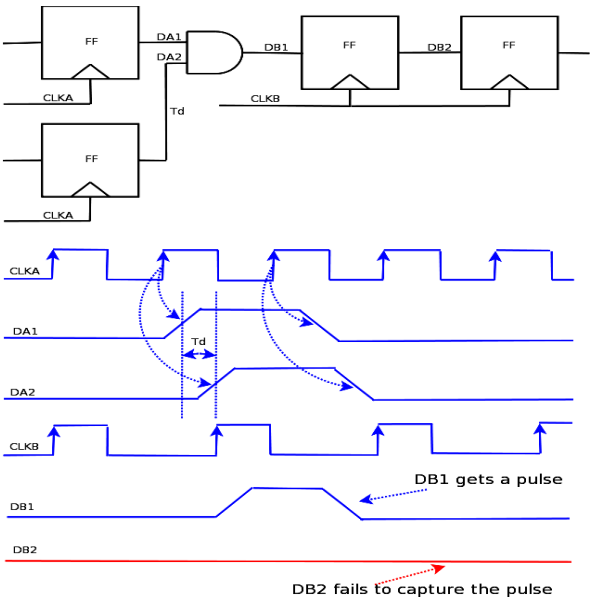

Figure 2. Convergence Issue

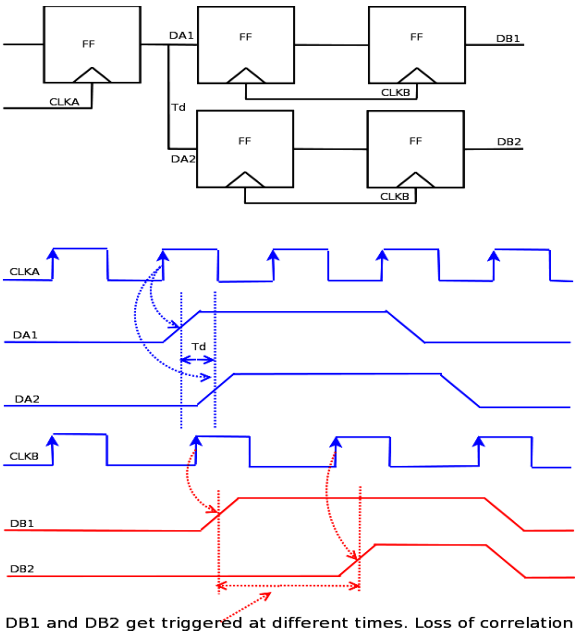

Figure 3. Divergence Issue

A CDC signal fed to multiple synchronizers also runs the risk of causing functional errors due to propagation delay in the cross-over path and differing metastable settling times of the different synchronizers [1]. There is a possible loss of correlation if each is intended to have the same value simultaneously as shown in Figure 3.

III. PROPOSED FLOW

We propose a methodical step-by-step approach for static CDC analysis that can be integrated easily in the sign-off timing verification flows of any design house without causing much churn. We have cited Synopsys PrimeTime [8] tool to illustrate the methodology. However, the illustrated approach is generic enough to work seamlessly with many other commercially available timing tools.

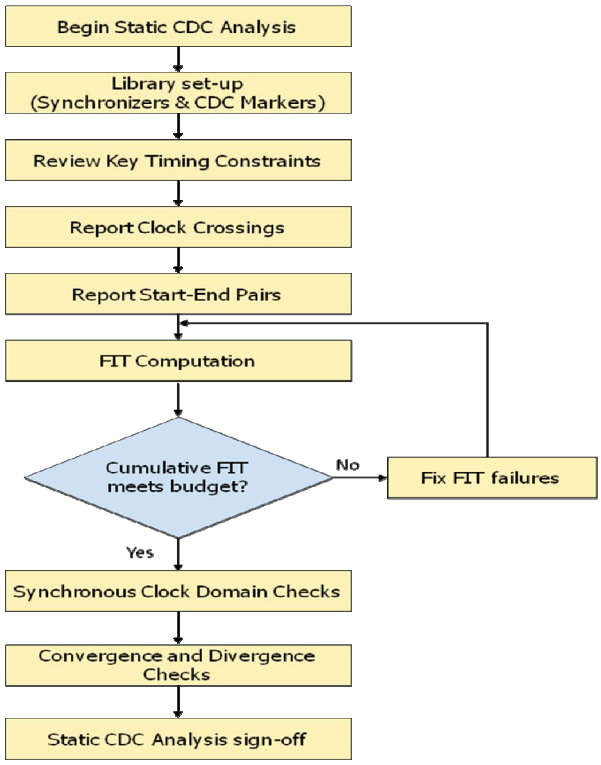

Figure 4. Static CDC Analysis Flow Chart

The following sub-sections describe the steps in our proposed methodology for static CDC analysis, which is summarized in the flow chart depicted in Figure 4.

A. Step 1: Library Requirements

Standard Synchronizers: The first step in our proposed flow is to include synchronizers as standard cells in a library rather than implementing these as cascaded stand-alone flip-flops in the design. This gives multiple benefits. First, and most important, it ensures easy recognition of synchronizers in STA and eases downstream checking. Standard-cell synchronizers also ensure the multi-stage flip-flops are placed together so they do not affect metastability. In addition, if the synchronizer structures are standardized, it becomes easy to

add and maintain dynamic features such as random delay modeling [4] that are needed for functional CDC verification.

CDC markers: In general, it is not easy for an STA engineer – or many industry-standard CDC verification tools - to distinguish control signals (for which synchronization is needed) from data signals (for which explicit synchronizers are not needed) in special synchronization schemes like Mux-D synchronization, hand-shaking, asynchronous FIFOs, etc. STA-based CDC analysis could flag hundreds of such data signals (for which explicit synchronizer cells do not exist) as missing-synchronization errors. One way to filter such false errors from the final results is to place uniquely named buffers, which we call CDC marker cells, at the output of the flop launching such data signals at the RTL stage itself. These CDC marker cells can then be identified in downstream STA runs. The CDC marker cells can also be deployed on signals that are essentially static and being launched from a very slow clock domain, which therefore may not need an explicit synchronizer. The fuse signals on a chip are an example of such static signals. Once used, the CDC marker cells can also serve the purpose of buffering any asynchronous CDC signals in the design. Given that fact, the additional impact on the area of the design with the insertion of these CDC marker cells is reduced to a minimum.

B. Step 2: Review of Key Timing Constraints

The most important requirement in CDC verification is to have all the clocks in the design correctly defined and propagated. Any sequential elements not receiving clocks would be skipped in downstream CDC analysis, so ideally, there should be no such cases.

Further, designs can operate under several modes. This is commonly reflected in STA with set_case_analysis, set_clock_groups, set_disable_timing, and other such constructs available in PrimeTime and other such timing tools. These constructs have the potential to mask the reporting of a CDC path, so it is imperative to understand these constructs well. One can either list the timing constraints in the form of a Synopsys Design Constraint (SDC) file from a timing session and review these constructs manually, or make use of sophisticated constraint analyzer tools such as Synopsys Galaxy Constraint Analyzer (GCA) [9], which provides an excellent cockpit for generating checks and reports for efficient constraint debug and analysis.

C. Step 3: Reporting Clock Crossings

We next enumerate the asynchronous clock crossings in the form of Start (Launch) - End (Capture) pairs for all the CDC paths that are topologically connected. However, timing tools run into a key limitation over here. These CDCs are declared as false paths and are therefore unconstrained, thus making the enumeration computationally expensive due to the need to trace many paths to sort by actual path delay instead of endpoint slack. We get around this issue by computing the intersection between collections of launch and capture points without computing the path delays.

For each pair of launch and capture clocks, we first create a collection S (startpoints) of all registers clocked by the launch clock. We then create a second collection E (endpoints) of all registers clocked by the capture clock. Now, for each startpoint in the collection S, we create a collection F (fanouts) comprised of all the registers to which the startpoint fans out through any combinational logic. We now need to ascertain which registers in the collection F are clocked by the capture clock under consideration. For that, we compute the intersection between the two collections E and F using Equation 2.

$$\text{intersection}\,(E\ F) = E - (E - F) \tag{2}$$

This gives us the listing of all the start-end pairs for that pair of a launch and capture clock. We repeat the same procedure for all pairs of launch and capture clocks to get the complete pair-wise list of all asynchronous startpoints and endpoints in the entire design.

D. Step 4: FIT Computation

The next step is to compute FIT for the enumerated CDC start-end pairs using Equation (1). This calculation will highlight any CDC in need of a synchronizer, or in need of a synchronizer of an increased depth, as having a large FIT number. FIT increases exponentially with an increase in capture clock frequency and decreases in supply voltage; therefore, it must be computed at the fastest clock frequency at the lowest voltage of operation. Given the fact that the incoming data rate r_{data} has a linear dependency on FIT, it can reasonably be approximated to 20% of the launch clock frequency for initial FIT calculations. Any high FIT numbers are fixed by either choosing a synchronizer of increased depth from the library for CDC, which are latency-insensitive, or by adjusting the incoming data rate to a number more accurately representing the design. Waivers are limited to high-speed multi-bit data CDC crossings, which are typically a part of FIFOs in which the synchronized control signals are used to transfer data.

Designers should also ensure that the cumulative FIT of synchronizers meets the target budget based on the product reliability perspective.

E. Step 5: Synchronous Clock Domain Checks

After going through the entire process of placing synchronizers of correct depths and CDC markers on asynchronous signals, it is advisable to perform a sanity check to confirm that no CDC markers and synchronizers got placed between synchronous clock domains. The check should also ensure that no false paths are specified between synchronous clock domains.

F. Step 6: Convergence and Divergence checks

The list of CDC startpoints (launch) and endpoints (capture) enumerated in Step 3 can be used to find any convergence and divergence cases. If the same startpoint is listed for multiple endpoints, then it's a case of divergence. Similarly, if there are multiple startpoints listed for the same endpoint, then it's a case of convergence. All such cases should be reviewed for possible issues.

IV. CASE STUDY

The methodical procedure described in Section III was evaluated on a 32 nm APU design. This design combined a multi-core central processing unit (CPU) with a graphics processing unit (GPU) together with several other IPs on a single chip.

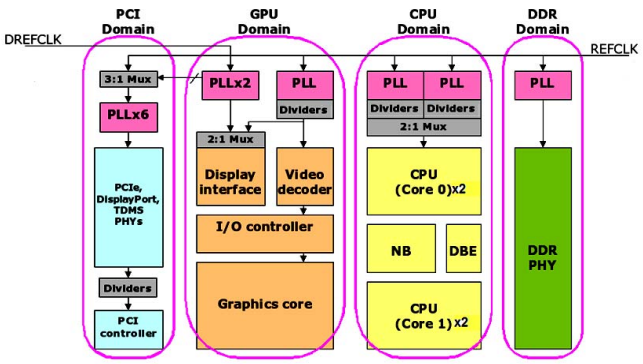

Figure 5. APU Design [10]

The number of transistors on the chip was close to a billion, and the number of clock domains was greater than sixty. A complete flat analysis was not possible on such a complex design. The methodology was therefore applied hierarchically in tandem with hierarchical STA flows. Static CDC analysis was first completed at the block or IP level and followed by verification at the SoC level. Single-clock domain IPs and IPs that did not have any untimed or unconstrained cross-clock domain paths at the interface were kept black-boxed at the SoC level. The rest of the IPs were grey-boxed.

The SoC-level grey-box timing netlist was built using an in-house utility, SoC-Extend. This utility reads the IP-level Verilog and Synopsys timing .libs. It then traces and captures the entire cone of data and clock logic leading to the first sequential elements present at the IP interface. The cone of logic subsequent to the first sequential elements is not captured because that is covered in block-level analysis. This hierarchical approach is used to obtain full-chip coverage.

Generating a grey-box netlist for designs like mixed-signal IPs that may not be completely standard cell-based might be difficult. In such cases, we need to enforce certain CDC rules at the IP interface such that the IP could be used as a black box at a higher level without leaving any coverage holes. These rules should prohibit usage of any combinational logic at the IP interface to avoid any convergence and glitch issues. Every asynchronous path at the IP interface that does not need a synchronizer should be marked with a CDC marker cell, and a design attribute on the IP interface pin should be used to indicate the same to the next-higher level in the design. Further, to avoid any divergence issues, the asynchronous signal should not fan out.

The entire process of static CDC analysis and verification on the APU took nearly twelve man-months. Given the complexity of the design, almost 20,000 CDCs across the entire chip had to be fixed to avoid seeing potential issues on silicon.

This approach can also be integrated with any RTL-level CDC verification flow [10]. Most structural CDC rules described in this paper are standard, and most of the RTL-level commercial tools have these rules built into their proprietary algorithms. Violation of any of the CDC rules usually gets documented in the form of a waiver. These RTL-level waivers can be translated into gate-level waivers with RTL-to-gate verification tools and applied to gate-level CDC results. The STA engineer can then limit his review to the remnants; if the resources and project cycle time are limited.

V. SUMMARY

Clock domain crossings are an inevitable aspect of modern designs with multiple asynchronous interfaces. The methodical approach illustrated in this paper integrates static CDC analysis with standard static timing analysis and constraint verification flow. Therefore, this approach does not cause too much churn in otherwise stable design flows. The illustrated methodology was successfully applied to a 32 nm APU design. The flow can be integrated with any RTL-level CDC verification efforts as well, and - when combined with functional CDC verification efforts - can significantly improve the overall timing and CDC verification coverage in the design.

ACKNOWLEDGEMENT

Sincere thanks for their help and support to several members of the CDC Working Group at AMD including Richard Bryant, Rajiv Hattangadi, Jim Montanaro, Aaron Grenat, Steve Kommrusch, Mark Silla, Robert Colyer, Cory Krug, Ryan Hyma, Priyank Parakh, Suresh Chodisetti, Erik Gonzalez, Chris Moench, Pete Hannan, and Mahesh Sharma.

REFERENCES

[1] Cadence, "Clock Domain Crossing," White paper. http://w2.cadence.com/whitepapers/cdc_wp.pdf

[2] S. Verma and A. Dabare, "Understanding Clock Domain Crossing Issues," EE Times 2007. http://www.eetimes.com/design/eda-design/4018520/Understanding-Clock-Domain-Crossing-Issues

[3] C. E. Cummings, "Clock Domain Crossing (CDC) Design and Verification Techniques using SystemVerilog," SNUG 2008.

[4] P. Parakh and S. Kommrusch, "A Smart Synchronizer: Pragmatic way to cross asynchronous clock domains," DVCON 2011.

[5] W. J. Dally and J.W. Poulton, "*Digital Systems Engineering*," Cambridge University Press, 2008.

[6] U. Ko and P. T. Balsara, "High-Performance Energy-Efficient D-Flip-Flop Circuits," IEEE Transactions on VLSI Systems, 2000.

[7] C. L. Portmann and T. H. Y. Meng, "Metastability in CMOS Library elements in reduced supply and technology scaled applications," IEEE Journal of Solid-State Circuits, 1995.

[8] Synopsys PrimeTime User Guide, Version F-2011.06

[9] R. Bishop, "A Case for Adopting Galaxy Constraint Analyzer," SNUG San Jose, 2011.

[10] P. Parakh and R. Sabbagh, "Achieving CDC Verification in the Billion-Transistor Chip Era," EDN, 2011. http://electronicdesign.com/article/eda/Achieving-CDC-Verification-in-the-Billion-Transistor-Chip-Era.aspx