Achieving ASIC Timing Closure with Speedcore eFPGAs (WP013)

Achronix
Data Acceleration

V1.1 – December 19, 2023 White Paper

The Achronix Speedcore eFPGA IP allows companies to embed a programmable logic fabric in their ASICs, delivering to end users the capability to modify or upgrade the functionality of an ASIC after being deployed in the field. This flexibility dramatically expands the solution space that can be served by the ASIC as it can be updated to support changing standards and algorithms.

Designing an ASIC with embedded FPGA technology requires using both ASIC and FPGA design tools and techniques. The eFPGA portion of the design must be validated prior to tape-out to ensure that the ASIC and FPGA portions of the design mesh well together and meet all timing constraints.

Timing closure is particularly challenging due to the fact that the eFPGA fabric may host any number of designs over the course of device operation. Each of those designs must work independently with the rest of the ASIC, and timing closure can only be said to have been met if all of the possible designs targeting the eFPGA fabric can meet timing.

Absent a well-thought-out flow, timing closure could be a struggle at best. For this reason, Achronix has developed a methodology that accommodates both the fixed nature of the ASIC and the programmable nature of the eFPGA. The purpose of this whitepaper is to describe how timing closure in this environment can be achieved in an efficient and straightforward manner. The Achronix approach to both static timing analysis and timed full-chip simulation removes the guesswork and frustration that would otherwise complicate timing closure.

The eFPGA Architecture: A High-Level View

The eFPGA fabric can be inserted anywhere inside an ASIC, meaning that individual eFPGA ports may connect to other blocks within the ASIC or to buffers connecting to external pins. There are two options for configuring each input to and output from the eFPGA block: they can either be directly connected or registered at the boundary of the eFPGA.

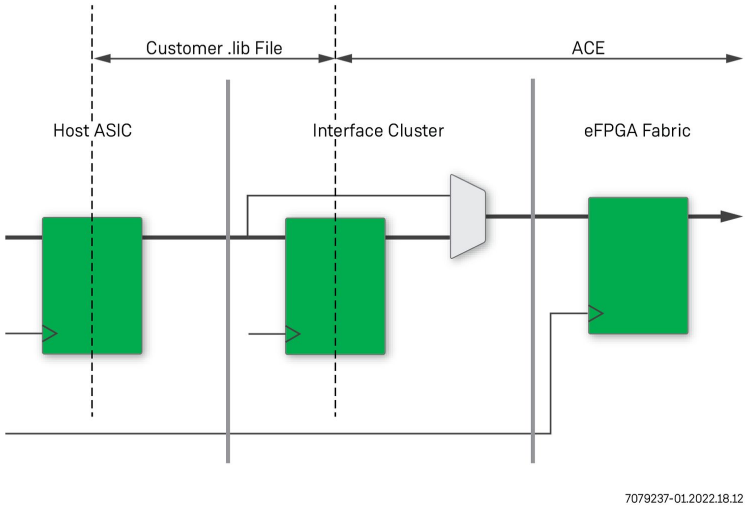

Figure 1 • Simple Timing Mode

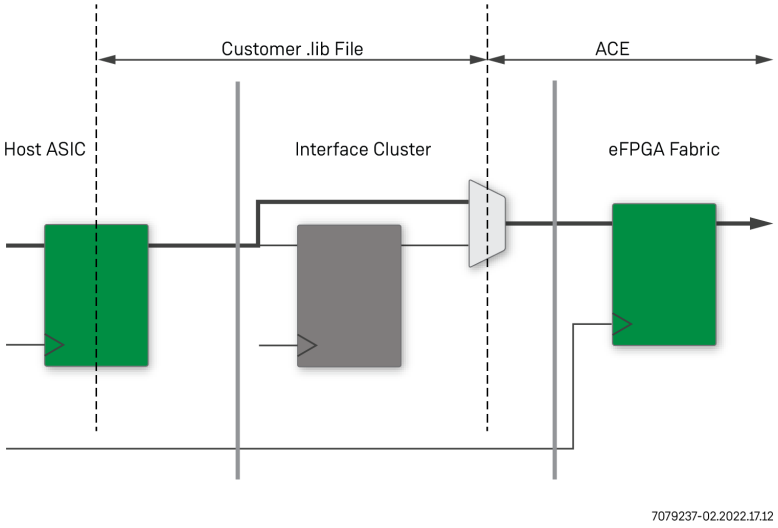

Figure 2 • Advanced Timing Mode

When registers are used, timing closure is simplified. ASIC signals can be timed independently up to the register; eFPGA signals can be timed independently from the other side of the register. The cost of this approach is that the register adds a clock cycle of latency to the signals.

Bypassing the registers in the eFPGA interface cluster eliminates the extra cycle of delay. However, achieving timing becomes more complex as the timing arc of the signal is shared between both the ASIC and eFPGA. The result is that the ASIC and eFPGA cannot be timed independently of each other.

This architectural choice results in approaching timing closure in one of two ways for each signal path – simple or advanced mode. With the interface cluster register in use, *simple mode* is employed; without the register, *advanced mode* is used. With simple mode, ASIC tools are used to close timing on the ASIC side; ACE tools close timing on the eFPGA side.

Advanced mode involves both ASIC and ACE tools. Scripts and utilities provided with the ACE tools help automate tasks necessary for stitching the two toolsets together in a way that closes timing on both the ASIC and eFPGA. Because the advance timing mode is where an unstructured approach can cause problems, this white paper focuses primarily on advanced mode timing closure.

Closing Timing with Advanced Mode

While a single engineer could implement both the ASIC and the eFPGA portions of the design, it is far more likely for these portions of the design are handled by different engineers/teams. In fact, the entire process of integrating an eFPGA into an ASIC involves a number of roles. For more details on the roles and responsibilities in an eFPGA engagement, see the blog post, **Who's Who in the Zoo**.

Typically, it is the back-end design team that is charged with closing timing between the host ASIC and the eFPGA. The flow for closing timing in advanced mode (see the following **figure**) proceeds as follows:

1. First, the back-end design team performs static timing analysis (STA) on the entire design, using the selected STA tool. The ASIC `.lib` file from the targeted foundry is used during this analysis. Separate runs are performed for each desired clocking scenario with the target frequency set to be as aggressive as possible.
2. The team then uses Achronix scripts to extract the ASIC and eFPGA portions of each signal entering or leaving the eFPGA.
3. These delays are provided as constraints to the ACE tools. Without these delays, ACE has no visibility into the timing outside the eFPGA block. The provided delays give ACE that visibility. If there are multiple applications intended for the eFPGA, each of these have a separate set of delays, and steps 1-3 must be done for each application independently.
4. ACE can now be run to confirm correct timing. Again, if there are multiple applications for the eFPGA, then each application is confirmed independently.
5. If timing passes for each design, then the process is complete. If timing does not pass, then one option is to remove some combinatorial logic from the offending path(s), and restart from step one. Another option is to return to step one, but either make the target frequency higher yet or add some margin to the affected signal(s). If there are multiple applications for the eFPGA, and if changes are required to the FPGA portion of the design to meet timing, only the changed application(s) must be rechecked. If changes are made on the ASIC side, then all applications must be rechecked.

A critical piece of the ACE tool suite from Achronix is a utility called SDCtimer. This utility runs with the STA tool based on Tcl scripts, generating the constraints for the eFPGA boundary timing. The actual delays calculated depend on the configuration and are determined by:

- Clock delays, margined at ±5% for on-chip variation
- Data delays, margined at ±5% for on-chip variation
- Setup times
- Hold times
- Pessimism parameters

All of the necessary components are provided by SDCtimer — none of this data is added manually by the designer.

The specific equations for timing depend on the relative configuration of clock and data signals. Examples include:

- Clock and data into the eFPGA
- Clock into, and data out of the eFPGA
- Clock out of, and data into the eFPGA
- Divided clock and data into eFPGA; data-in clocked by a divided clock on ASIC side
- Divided clock and data into eFPGA; data-in clocked by an undivided clock on ASIC side
- Inverted clock and data into the eFPGA
- Clock into the eFPGA on more than one clock signal

ACE automatically handles the assembling of the appropriate delay components for each of the above scenarios, as well as the many other variations that are possible. SDCtimer also handles pin renaming as necessary when moving between the ASIC and ACE tools.

Many modern designs have multiple operational modes. It is best to exercise the modes independently in the eFPGA in the same way that those modes are verified in the ASIC portion of the chip. For the most part, it is recommended to use the same modes in the eFPGA verification as are used for ASIC verification.

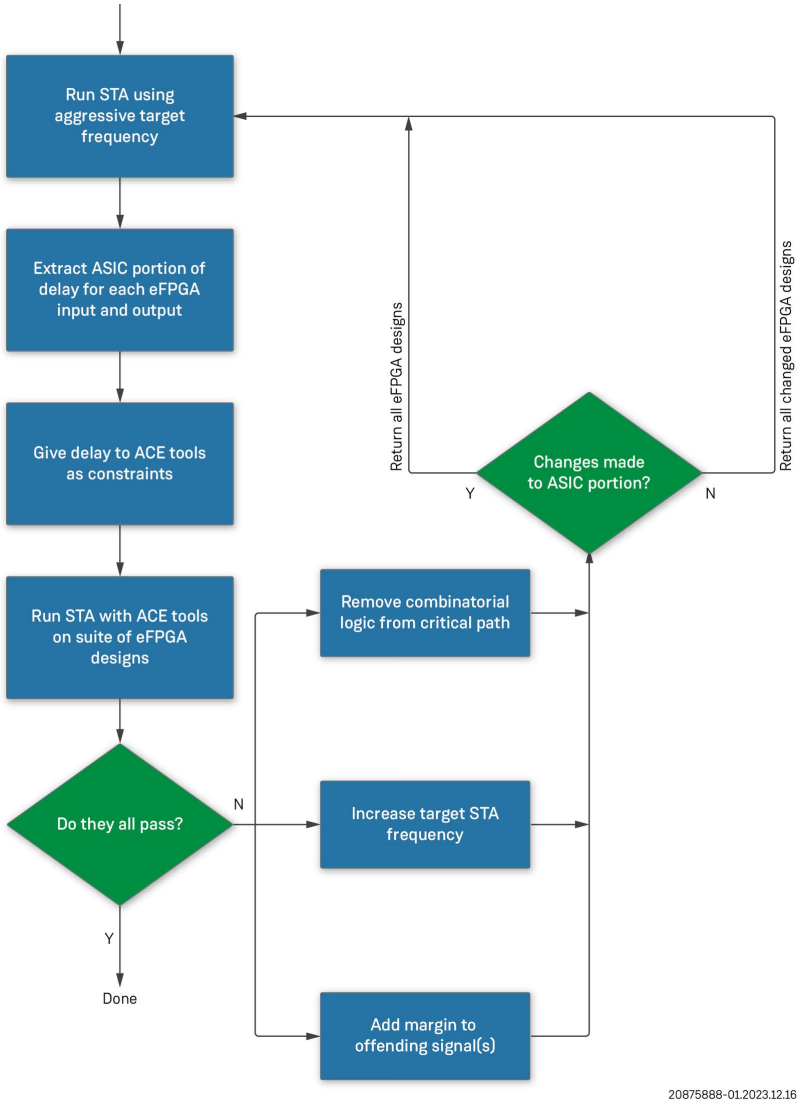

Figure 3 • Timing Closure in the Advanced Mode

Efficient Timing Closure for Both ASIC and eFPGA

Achronix's timing closure methodology provides a straightforward way to time both the Speedcore eFPGA and ASIC portions of a chip. This methodology leverages both standard ASIC tools and the ACE tools provided by Achronix. This process enables both static timing analysis and full-chip timed simulation.

Timing can be confirmed across process and operation corners, for different operational modes, and even for multiple designs that reside in the eFPGA. Design teams can have confidence in the results once the tools say that the timing requirements have been met.

For more information on the Achronix Speedcore eFPGA blocks and the ACE tool suite, visit **www.achronix.com**. For more details on timing closure with Speedcore eFPGAs, refer to the ***Speedcore ASIC Integration and Timing User Guide*** **(UG064)**.

Achronix Semiconductor Corporation

2903 Bunker Hill Lane
Santa Clara, CA 95054
USA

Website: www.achronix.com
E-mail : info@achronix.com

Notice of Disclaimer