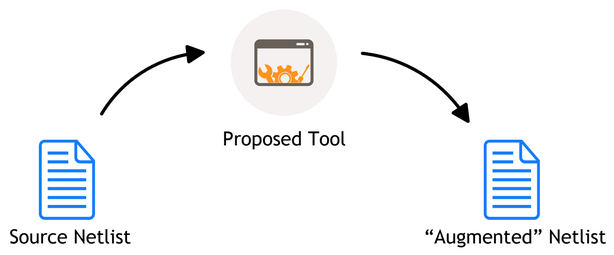
Formal Verification of Clock Domain Crossing using Gate-level Models of Metastable Flip-flops

Ghaith Tarawneh*, Andrey Mokhov and Alex Yakovlev
Newcastle University, UK

The Basic Idea

This is a demo of a tool for verifying multi-clock designs.

The tool transforms a gate-level netlist into an "augmented" netlist which is capable of reproducing many of the problematic phenomena at clock domain interfaces in digital simulation.



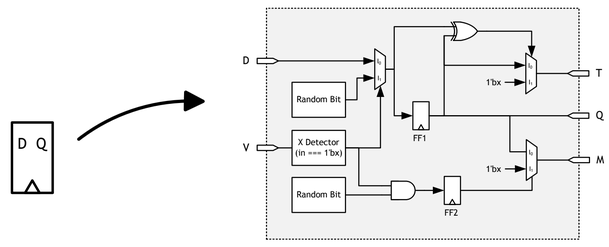
- Contains some CDC bugs (e.g. metastability, non-deterministic crossover latencies, glitches)
- CDC issues cannot be reproduced in digital cycle-based simulators (e.g. Modelsim)
- CDC bugs can therefore escape conventional testbench and formal verification but manifest in silicon

- Contains additional components to reproduce CDC issues digitally.
- Can be passed to formal verification tools or used in testbench verification to identify and debug CDC issues in a very simple and intuitive fashion.

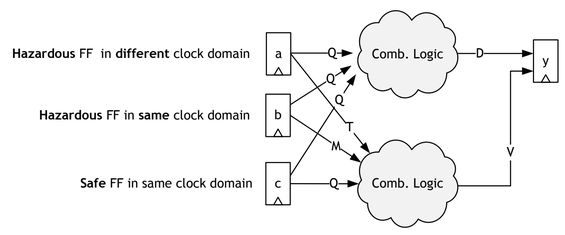
How does it work?

The netlist transformation applied by the tool consists of two steps:

1. Flip-flops are replaced with models of metastable FFs that can simulate (1) setup/hold time violations, (2) non-deterministic inputs/outputs and (3) prolonged clk-to-q delays.



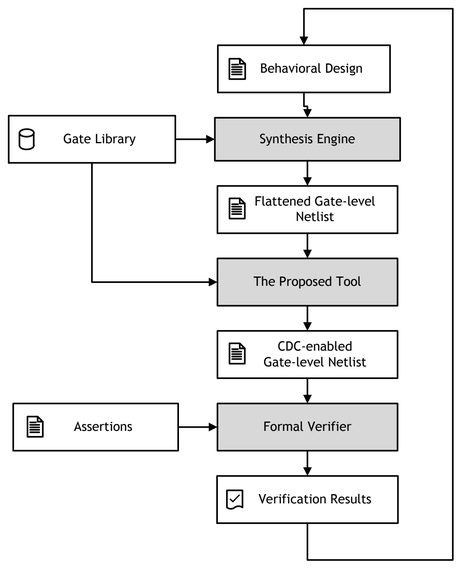
2. Additional combinational gates are added to simulate the transfer of timing violations between FF models.



Path Sensitization Detector

Proposed Flow

The tool can be combined with formal verification to create a potent CDC verification flow. Augmented netlists can be passed to formal tools to identify and debug any issues emanating from the abnormal behavior of signals and flip-flops at clock boundaries.

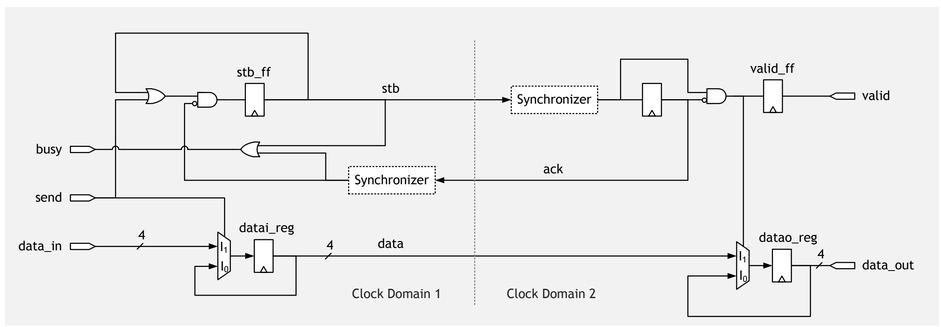


Case Example and Verification Results

The following sender-receiver circuit is used to showcase how the tool works. The sender and receiver components are in different clock domains and communicate via a handshake.Three SystemVerilog assertions were specified to ensure that the circuit works correctly. Both source and augmented circuit netlists were passed to a formal tool to check for violations of these assertions.

How was this circuit tested?

We synthesized and verified this circuit without and without sender and receiver synchronizers (i.e. sender only, receiver only, both or none).
In each case we passed both source and augmented netlists to a formal tool (Incisive Formal Verifier) to check whether any of our assertions can be violated.



Synchronizer(s)	Assertion	Status (✓= pass) Source Netlist	Processed Netlist
None	as_correct_transfer	✓	-
	as_sender_handshake	✓	-
	as_no_blocked_transfer	✓	-
Sender Only	as_correct_transfer	✓	-
	as_sender_handshake	✓	✓
	as_no_blocked_transfer	✓	-
Receiver Only	as_correct_transfer	✓	-
	as_sender_handshake	✓	-
	as_no_blocked_transfer	✓	-
Both	as_correct_transfer	✓	✓
	as_sender_handshake	✓	✓
	as_no_blocked_transfer	✓	✓

as_correct_transfer: when *valid* is asserted, the value of *data_out* must equal the value of *data_in* when *send* was last asserted

as_sender_handshake: *busy* must be asserted after *send* by 1 cycle

as_no_blocked_transfer: any assertion of *send* must be followed by an assertion of *valid* in finite time

Verification results:

The formal tool correctly identified CDC issues when the circuit netlist is augmented by our tool (but not when running on the source netlist). Note that neither our tool nor the formal verifier attempt to recognize synchronizers or have any coded knowledge about what synchronizers are or what they do. The lack of synchronizers was nevertheless correctly identified as a problem because, in the augmented netlist, scenarios were identified in which metastability occurred and caused functional differences in the circuit's behavior that violated some of the assertions.

Why is it useful?

Verifying multi-clock designs is **notoriously difficult**.

Most CDC faults are caused by analog phenomena(e.g. flip-flop metastability) that most digital tools **cannot simulate**. Conventional CDC verification tools attempt to make up for this by checking designs using structural and functional "rules of thumb". This approach is safe but has a number of limitations (see box titled "Advantages …").

The proposed tool addresses the fundamental issue at the heart of CDC verification difficulties: the unobservability of CDC faults in digital simulation. This makes CDC issues as easy to identify and debug as their synchronous counterparts.

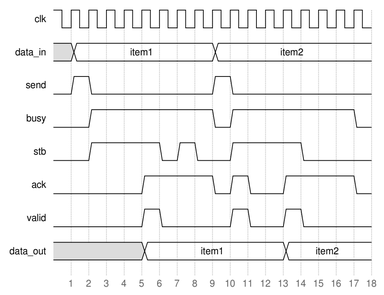
Advantages over state-of-the-art commercial CDC verification tools

The proposed tool:

- Reports fewer false positives
- Requires no knowledge about what synchronization scheme is used or how clock domain logic is supposed to work
- Capable of verify non-stereotypical designs that don't adopt conventional CDC design principles
- (With formal tools) can generate counter-examples showing CDC faults and their effects in simulation waveforms
- Can be used to determine what types failures can result from CDC faults (e.g. invalid state transitions, data corruption)

CDC Design Puzzle

To our surprise, the tool identified a counter-intuitive way for the circuit above to fail. Can you guess how a missing **sender** synchronizer can cause incorrect data to be received? The solution is below.



Contact: Ghaith Tarawneh (**ghaith.tarawneh@ncl.ac.uk**) ● For more details please refer to our paper with the same title in the proceedings of DATE2016 ● This work was supported by EPSRC grants EP/L025507/1 (A4A) and EP/J008133/1 (TRAMS2).