Avoiding Confounding Configurations

An RDC Methodology for Configurable Designs

Eamonn Quigley, Jonathan Niven, Kurt Takara
Arm Limited, Cambridge UK, Siemens Digital Industries Software, Fremont CA

1. Challenges

A. Asynchronous data or signal crossings are subject to corruption
- These are known as Clock Domain Crossings (CDC) or Reset Domain Crossings (RDC).
- The challenges of CDCs are understood but those of RDCs are often overlooked.
- The corruption is due to a property of a sampling device, known as metastability, that may result in intermediate (non-binary) values for a non-deterministic time after the asynchronous event.
- Safe design practices are known and tools such as CDC or RDC verification verify the presence, and often the correct behavior of synchronizers implementing these best practices. These can be safely verified in simulation.

B. Some RDCs are also CDCs
- Duplicate review and debug effort occurs when CDC and RDC paths overlap.
- Parallel CDC/RDC runs accelerate compute efforts at the cost of human effort.
- For example, the asynchronous domain boundary for clocks and resets is the same for this synchronizer with combinational logic – the CDC is also an RDC.

C. Highly configurable designs can require many CDC (and RDC) verification runs
- Configurable number of sub-blocks per slice, slices per IP
- Number of CDC and RDC groups vary with configuration
- Potential for cross-slice interactions
- All combinations should be considered across CDC and RDC groups
- Number of crossings to verify is variable but substantial
- Crossings consist of more than one signal
- Reviewing violations is complex
- Some violations are common between configurations
- Others are unique

D. Constraints are not sufficient in a highly configurable design
- Constraints define fixed properties, assumptions, operational space
- Waivers post-process results where a reported issue isn’t a problem
- For highly configurable designs, constraints cannot completely describe the operational space to preclude all waivers

E. Waivers must be preserved through hierarchy and configurations
- CDC (and RDC) are typically run low-level first, then simple high-level, then complex configuration-high level
- Issues are missed when flexibility is added with overly-restrictive wildcard-based waivers
- Un-anticipated waiver/constraint warnings occur when a superset of all precise waivers are created and used

2. Solutions

A. Eliminate redundant verification by utilizing commonality between CDC and RDC verification
- Use common configuration and status processing found in some tools such as Questa CDC and RDC.
- Verify CDC design and constraints first - CDC constraints are required for RDC analysis
- Then use clean CDC design and constraints, focus on RDC
- Leverage Questa’s identification of joint CDC/RDC paths to defer to CDC, thus focusing on pure RDC

B. Enable Continuous Integration and manage result lifecycle and commonality using tool status flows
- Utilize the Questa constraint and status flows to track issues from start to completion
- Configurability generates a large amount of issues that we must manage efficiently
- Embed messaging scheme below into waivers to accelerate reviews by grouping similar paths and tracking

3. Results

A. Conditional Tcl waiver loop example

- Avoid the use of wildcards to cover instantiation hierarchy
- Avoid the use of module-based waivers due to configuration-based inter-slice domain crossings
- Generated precise waivers for any configuration based on Tcl loops
- Also expanded waivers to use the embedded messaging comments for each waiver

B. Python CI example

- Conditional Tcl waiver loop example

- Python filter-crossing script example

4. Summary

RDC analysis is made more complex by configurability and scaling, so a different approach that is focused on efficient closure in such an environment is necessary. For complex designs, a Continuous Integration flow is required to verify and manage the large number of CDC/RDC paths and violations. By first eliminating all sources of issues but true RDCs, the team focuses specifically on RDCs and their status flows and ability to identify CDC/RDC path overlap, the team can triage issues quickly and reduce redundant effort. The use of the embedded message techniques allowed both waiver grouping and correlation as well as bug ticket correlation that in resulted in CI efficiencies and improved design review productivity. Finally, structuring and applying waivers in loops allowed for straightforward adaptation to the configurability in the design.

References