2025 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION INDIA

Next-Generation CDC and RDC Closure: An AI/ML-Driven Approach for Automated and Validated Constraint

Abdul Moyeen
abdul.moyeen@siemens.com
Siemens EDA
Fremont, CA 94555 USA

Ravi Pathak
pathak.ravi@siemens.com
Siemens EDA
Bangalore, Karnataka, 94555 INDIA

***Abstract*- In this paper we propose a flow where we use powerful AI/ML based utility 'CDC/RDC assist®' available with Questa CDC® to generate a set of CDC/RDC constraints and then use 'protocol/constraint validation' flow to formally and functionally validate those generated constraints. Theoretically, we suggest a proven automated flow to generate a set of validated CDC/RDC constraints without user intervention.**

I. Introduction

For accurate results, CDC/RDC analysis requires a good setup. This setup provides the tool specific constraints required to achieve accurate CDC/RDC closure within a reasonable amount of time. What we want to achieve is accuracy balanced with tool performance. You cannot be too optimistic and allow CDC bugs slip by also cannot be over pessimistic and delay your project. Normally CDC/RDC flow depends a lot on manual analysis and fixing of CDC setup, it takes significant time and energy from the design team bandwidth. Our proposed solution is to automatically generate the needed constraints which are proven to be correct for your design. As the flow is fully automated, it needs no human intervention, hence help achieve quality RTL faster.

II. Proof of concept

To limit the scope of this paper, as proof of concept, we have only created the automated flow for the "syncenable" constraint which is used to specify the enable signal for control-based crossings.

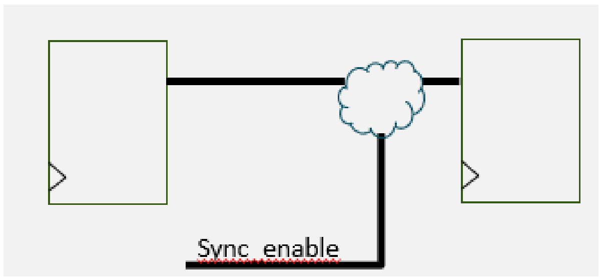

Figure 1. Generic CDC crossing where data capture is controlled by a sync_enable

Sync enable are essentially isolating the TX data while it is changing hence blocking metastable signal to be captured by RX flops. The Concept can easily be extended to other constraints supported by existing flows. With future enhancements, we can move closer to achieve constraint less CDC closure where user is not expected to create major part of CDC/RDC setup constraints. It will require less manual effort, intelligent constraints generation via RTL and less waiver dependency.

CDC assist® : This is a powerful AI/ML based utility available with Questa CDC® which can analyze the available CDC results and provide some suggestions to fix the gaps in CDC setup. This helps achieve a faster closure for a good CDC/RDC setup. These suggestions are mainly constraints which can be then applied to existing CDC/RDC setup in iterative process until you achieve desired results. Along with other supported constraints, Questa CDC Assist, with its light-weight formal capability, can recognize control signals which are possibly controlling the capture of CDC signals on the receiving register. The user is required to review these suggestions before feeding them back into the CDC run.

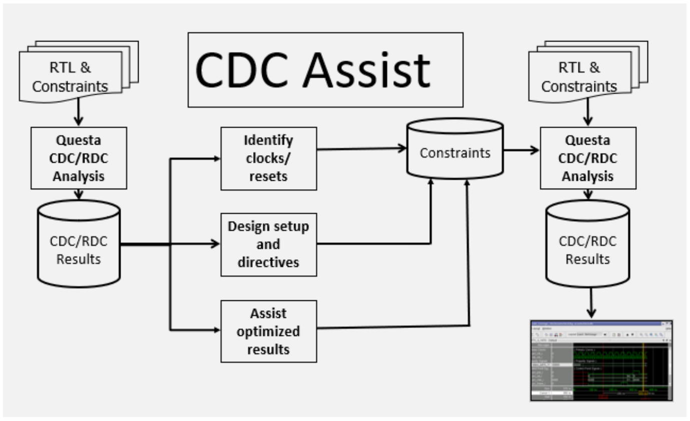

Figure 2. CDC Assist® Flow

Quest Protocol Validation flow® : This is a push button flow which automatically generates SV assertions for a list of CDC constraints and structures, including syncenable based crossings. Along with these assertions, a fully automated setup is also generated which help user to run the generated assertion in a formal flow and validate those assertions formally. Assertions which are not proven formally are then passed on to simulation setup.

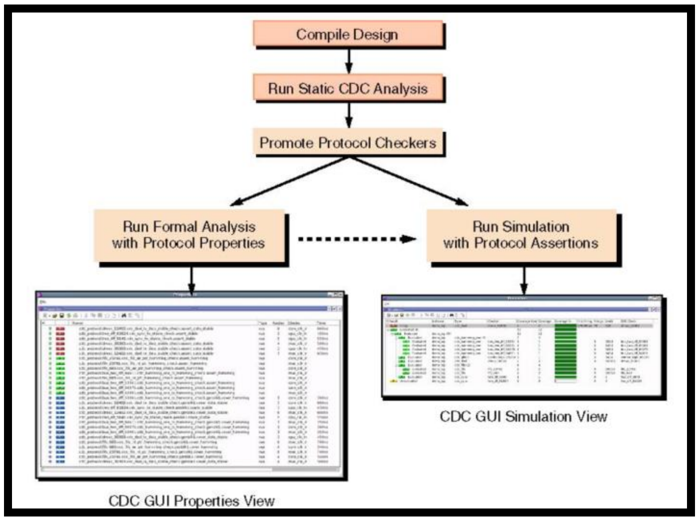
Compile Design
Run Static CDC Analysis
Promote Protocol Checkers
Run Formal Analysis with Protocol Properties
Run Simulation with Protocol Assertions
CDC GUI Simulation View
CDC GUI Properties View

Figure 3. CDC Protocol Validation flow

As part of the suggested flow in this paper, we initially generated sync_enable constraints with the help of CDC Assist utility. Some TCL scripting was used to automate the flow to feed constraints generated by CDC assist after minor formatting back into the protocol flow for validation.

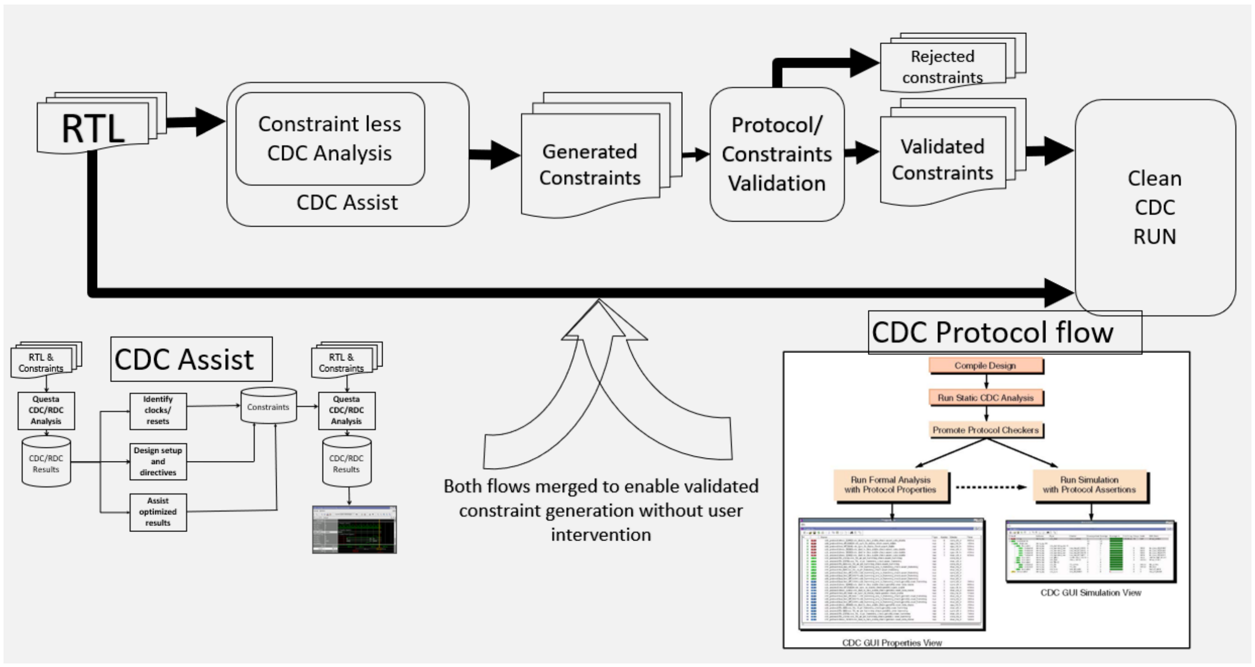

Figure 2. Proposed Flow to automatically generate proven CDC constraints.

PROPOSED FLOW

- We started by taking a constraint less CDC run or design with available half-cooked CDC setup. For this paper, 1 design used no initial constraints except for DFT mode selection, and other two designs used available SDC for clock definitions.
- The CDC Assist flow was enabled with CDC analysis.
- ‘sync_enable’ constraints are filtered out from generated suggestions from CDC assist report. These are added to CDC constraint file.
- Another CDC run is fired with these additional constraints with protocol flow enabled.
- Generated assertions and checkers are then run with generated Makefile in formal flow.
- Failing ‘sync_enable’ constraints are filtered out of original constraint file and reported as a separate report.
- A new constraint file is created including only passing ‘sync_enable’ constrsints

III. RESULTS

TABLE I
RESULTS FOR CANDIDATE DESIGNS

Design	**Number of CDC Paths**	**Suggested Syncenables**	**Proven Good Syncenables**	**Proven false Syncenables**	**Unused/Unproven Sync enables**
Design1	~200k	325	250	63	12
Design2	~350k	107	70	23	14
Design3	~150k	40	25	15	0

It can be seen in the results below for some candidate designs, the raw syncenable found were formally proven to be either true or False.

Number of CDC Paths	**:** This indicates the complexity of the design.
Suggested Syncenables	**:** These are number of sync_enable constraints generated by CDC assist®
Proven Good Syncenables	**:** These are formally proven good sync enable constraints where suggested rx_enable is indeed controlling the data capture on RX side and there is no change on TX register while enable sinal is active.
Proven false Syncenables	**:** These are failing sync_enable constraints and problems in design. These should be fixed in the design. These are cases where at least one control signal is enabling metastable signals to be captured by RX flop.
Unused/Unproven Sync enables	**:** These are either vacuous cases or cases which are unproven due to formal capacity. Designers should either update the testbench to provide simulation scenario to prove this in simulation or confirm if the functionality is out of scope of design operation.

The results provide us with the following conclusion:

- Quick closure of valid syncenable recognized in the design.
    - Formal property time limit was 2 hours per property, after that it was taken to simulation.
- Expected failing syncenable without user intervention.
    - Apart from passing proven sync_enable, we also got failing sync_enables which points to unwanted signals in input cone of data isolation circuitry.
- User can now proceed to fix the design for all the proven failures.
- Even the unproven or unused cases give us insight for any unknown signal controlling the data capture.
    - These are either vacuous in formal or uncovered in simulation.

IV. Summary and future work

We were able to achieve a list of all validated and formally proven sync-enables in the design. Every constraint is formally or functionally validated. The same technique can be very easily applied to other supported constraints without even changing the scripts. Manually, this task can take multiple iterations/days. As automated, this completed in two runs. The starting point does not require a mature CDC design, so faster CDC/RDC closure can be achieved even on a large project. For future work, the same flow can be explored for all available constraints in Questa CDC Assist. We can even go further and increase the scope of Questa CDC Assist and protocol flow based on this proof of concept.

References

[1] Abdelouahab Ayari, Sukriti Bisht, Sulabh Kumar Khare and Kurt Takara "Don't Forget the Protocol! A CDC Protocol Methodology to Avoid Bugs in Silicon" DVCon Europe 2017.

[2] Sukriti Bisht, Sulabh Kumar Khare, and Ashish Hari "A Systematic Take on Addressing Dynamic CDC Verification Challenges" DVCon United States 2019.

[3] Abdul Moyeen and Manish Bhati "Empowering CDC analysis methodology with root cause analysis" DAC 2024.