Formal based Automation Framework to Verify Coherent Connectivity of Multi-instance IPs

Somesh Gode, Abhinandan Sharma, Vinod Paparaju, Malav Shah

59 DESIGN AUTOMATION CONFERENCE
FROM CHIPS TO SYSTEMS — LEARN TODAY, CREATE TOMORROW

1. Motivation

- Consider a SoC under design, there is a good chance it has at least one IP which is instantiated multiple times.
- Although the instances are connected with the SoC in a similar fashion, a design verification engineer spends time and effort in verifying the integration of all those instances as per specification, individually, through multiple functional test cases.
- In a new spin-off of this SoC, if the number of the IP instances is higher, the DV engineer would expand the testplan to add more tests for these new instances, with test sequences in some cases could be broadly similar to what is being run on existing instances.
- When we reviewed the testcases and analyzed the resultant coverage caused by them across IP instances, we were able to observe a remarkable similarity among the connectivity of different instances of an IPs with rest of the SoC.
- As the testcases targeting a set of connections are similar for all the instances of an IP, we explored to understand if we can optimize this. It brought us to the question
  - **Can we verify one instance of an IP and use that information to verify other instances quickly without many functional testcases?**
- We came up with a generic and reliable approach that can match the coverage requirements, and achievable with our existing tools.

2. Proposed Solution

Approach	Details on how?
Verify one of the instances and understand how it is connected to rest of the SoC	How to understand the connectivity of an IP with rest of the SoC? • We have used the formal verification approach. • Reverse Connectivity feature in Cadence Jasper CONN App is used to generate connection files of a target.
Predict how other instances should have been integrated into the SoC	• We now have a comprehensive document of where all connections to IP are coming from and going to. How to predict connections of other IP instances with this info? • We have created Python based automation scripts that can analyze a connection and predict a similar mapping for the new IP instance in a scalable way.
Are these predictions reliable?	• We have come up with different approaches to make the predictions as accurate as possible.
Verify that our expectations match the implementation.	We now used the automation to create connection files for all the instances of the IP. **How to verify them?** Can we do it at once? • We have created a comprehensive tool which takes in all the files and performs the connectivity checks in a single step.

3. What can be achieved with this?

1 — Verify connections in the current RTL release with an older release as a benchmark (preferably after the IP is fully integrated)

- Generate connection files of previous release
- ⇩
- Run them as it is on the newer release

2 — Verify the connections for all instances of an IP using one or more of those instances (Golden instances) as the benchmark.

- Generate the connection file for a golden instance
- ⇩
- Scale it to predict the connection files for remaining instances
- ⇩
- Then run them on the same RTL version

3 — Verify the connections in the new SoC variant (different number of IP instances) with an older SoC variant as the benchmark.

- Take 1 or 2 connection files of the old version
- ⇩
- Make small edits according to the new version
- ⇩
- Scale it up to get the connection files of remaining instances
- ⇩
- Then run it on the newer version

4. Digging deeper – Preparing for Prediction

Conventional Reverse Connectivity

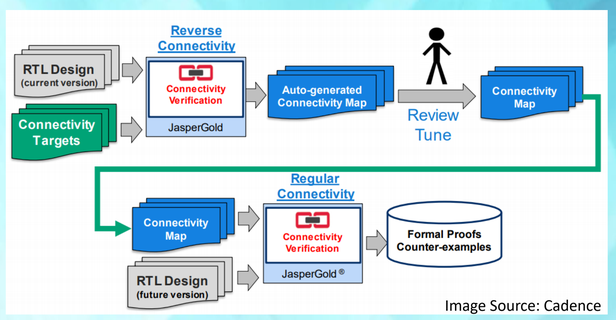

Image Source: Cadence

- The reverse connectivity techniques used are already part of the Cadence Jasper CONN app.
- However, the practice of prediction for the connections is not currently used or documented in any known sources.

Leveraging Golden Instance

- Our first step involves understanding all the connections of an IP instance. A typical connection has multi-line information, where the first line informs us of the source and destination and the following lines contains additional information of the connection.
- Using the depth and complexity as variables, we have done experiments to understand what will be the most appropriate combination to use for reverse connectivity, using an instance of an IP in our SoC as Target.
- For an effective connections list which focuses on the integration of IPs in the SoC, we have blackboxed the IPs (as internal connections already verified in IP level DV).

5. Automation to generate connection files

- We have developed an automation tool in Python programming language which can take a connection file of an IP instance and generates the connection files for other instances using prediction techniques.

MODE 1:

- ONE connection file is taken as input and ONE connection file (per instance) is generated as output.
- Connections like clock or reset are fairly easy to predict but it is challenging to predict few peculiar type of connections. This would result in inaccurate predictions which would cause some of the properties to fail and requires additional effort to correct it.

MODE 2:

- Minimum of TWO connection files are required as input and we generate TWO connection files as output for each of the requested instance.
- By doing pattern matching from the two inputs then proceeding to the prediction, we are able to establish excellent confidence in the predicted connections and separate out the connections that we cannot reliably predict.

	IP-1 Input Instances: 0, 16 Output Instance: 8 Total assertions : 608	IP-2 Input Instances : 0, 9 Output Instance: 4 Total assertions : 391	IP-3 Input Instances : 0, 3 Output Instance : 2 Total assertions : 53
MODE 1	conn_file.csv No. of assertions : 608 No. of proved assertions : 453	conn_file.csv No. of assertions : 391 No. of proved assertions : 345	conn_file.csv No. of assertions : 53 No. of proved assertions : 53
MODE 2	conn_file.csv No. of assertions : 560 No. of proved assertions : 556 exc_file.csv No. of assertions : 48	conn_file.csv No. of assertions : 369 No. of proved assertions : 368 exc_file.csv No. of assertions : 22	conn_file.csv No. of assertions : 53 No. of proved assertions : 53 exc_file.csv No. of assertions : 0

- Our automation also uses “Save and Restore” feature of Jasper tool to create snapshot of elaborated design to save execution time of the reverse connectivity file generation and connectivity proofs.
  - On the first run on any RTL version, the elaborated design is saved in a common folder. This will take significant time depending on the design.
  - For any further runs on the RTL version by any user, Automation restores the stored elaborated design and desired operation will be initiated in 10 – 15 secs, a 50x improvement over normal run time.

6. Impact & Conclusions

- This flow is currently in use for our SOCs.
- Helps in early detection of issues, which would help to improve the DV timelines.
- It also offers comprehensive coverage, not leaving out any connection, hence its usage enhances the confidence in the final RTL release.
- Prediction capabilities, optimizations and Jasper integration helps us in reducing the effort required to be spent for the execution.
- This flow is generic in nature, scalable and can be leveraged widely for any SOC DV with multiple IP instances.

Texas Instruments