2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION INDIA

Accelerating the SoC Integration Verification Cycle Time Leveraging the Legacy Design Confidence

Abhinav Parashar, Prasanth Kumar Narava

***Abstract*-In a traditional SoC Verification flow, whenever the integration of an SoC changes it takes several days of effort in order to identify the changes by setting up the DV Testbench environment, running simulations and regressions. Since multiple incremental SoC releases with integration changes are expected till a stable and bug free SoC release version is available, the effort required for finding connectivity changes between each subsequent release gets multiplied. Hence there is a huge bring up cycle gap in SoC design cycle time between SoC integration and DV closure.**

When an SoC is derived with cut-down or additional features from a silicon proven device, the verification cycle time for the spin off device is expected to be reduced to bring up the silicon readiness to market faster ensuring the legacy features remains intact. The above approach mentioned for verification was time consuming. We need to have faster solution for leveraging the silicon proven connectivity in the spin off devices.

The Reverse Connectivity approach can be used to solve the two problems stated above. This Solution helps in the development of derivative designs with shorter time frames where connectivity verification of SoC is the key start point, also ensuring the quality of design with minimum efforts. This Methodology also helps to give the feedback of the connectivity changes within the SoC design releases. The feedback hence obtained is used to identify the issues due to integration changes very quickly without an actual Testbench bring up.

I. INTRODUCTION

The connectivity verification of an SoC plays a key role in an overall SoC design cycle. The critical functionality of an SoC might get impacted if the connectivity in a SoC is not proper.

In a typical SoC design cycle there exists multiple iterations of Integration changes after incorporating the bug fixes which were reported in the previous SoC design release. In a traditional flow to verify these changes ample time is being spent in bringing up the testbench, running simulations, regressions of test suite and then debugging the failures. This effort gets multiplied when there are multiple SoC releases until a stable and bug free release comes up. Here there is a huge bring up cycle gap between SoC integration and DV closure when we have changes in SoC Integration in Incremental SoC releases.

In a typical SoC design platform we might have multiple spin-off devices having add-on or cut-down features derived from a Legacy silicon proven device. For a faster silicon readiness to market of a spin-off device the verification cycle time is expected to be reduced. The former approach discussed above is time consuming and not robust to meet the RTM timelines. Also it doesn't give confidence on legacy features which are intact.

Each spin off device in a platform can possibly have paper spins and in turn have multiple incremental releases until a stable release. There will be a high chance for connectivity misses and impact on baseline features. There is no Formal based methodology to make this solution faster, robust and reusable.

In this paper we have proposed a Reverse Connectivity based solution which will help in faster findings of connectivity changes or misses between incremental releases of an SoC along with gaining confidence on the legacy features between all spin off and paper spins in a given device platform.

II. EXISTING LITERATURE

Existing verification approach involves the simulation based DV to target the connectivity related changes which is time consuming and not robust. Simulation based connectivity checks will be limited to the designs like Interconnect IPs between domains, IP for pad connections, IP for configurable event connections which makes this approach very conservative.

There is no readily available formal based approach to make the intent of checks complete and faster to ensure all baseline features in spin-off devices were intact and correct.

III. CONCEPT OF REVERSE CONNECTIVITY

Reverse Connectivity is an approach through which the connectivity information of a silicon proven device can be extracted through Reverse Connectivity app of Cadence- Jasper Gold tool and the information hence obtained can be used to compare and identify the connectivity changes in subsequent SoC or derivative SoC releases.

Initial reference design is a Silicon proven device from which the Reverse connectivity map consisting of connectivity checks are generated and when proven on same golden design, all checks should pass. The Reverse connectivity map hence generated has all the baseline connectivity information, now will be used to prove the new design with add-on or cut-down features. The checkers result in either pass or fail. The failures will be due to the connectivity changes or valid bugs which can be debugged and reported as RTL changes. Fig.1 shows the flow of Reverse Connectivity approach.

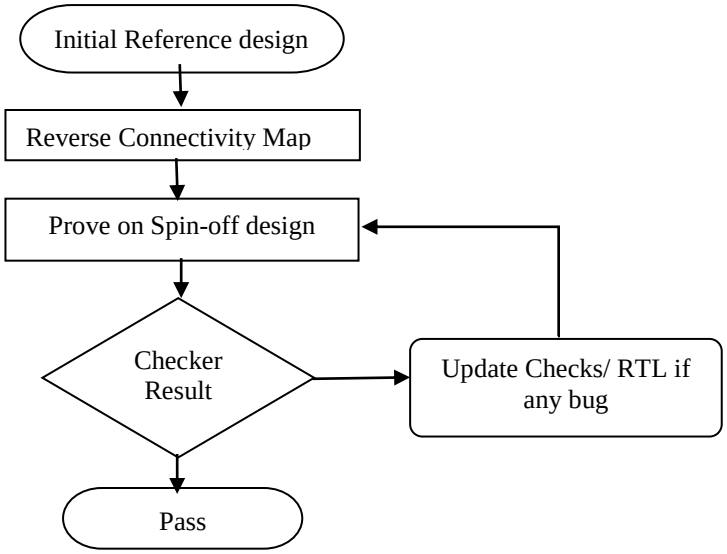

Figure 1. Flowchart representing the methodology of Reverse Connectivity

IV. EXECUTION FLOW METHODOLOGY

Whenever there is a new SoC design release, it is passed through two phases of verification flow. Phase 1 DV flow ensures all the baseline features were intact and gives a confidence on legacy features. Phase 2 DV flow ensures all the connectivity changes when compared to previous SoC release were intact or some expected changes exist due to bug fixes. Fig.2 represents the overall flow of the solution proposed for connectivity verification.

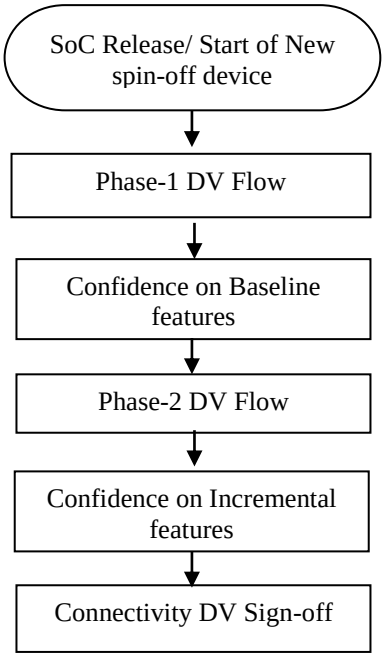

Figure 2. Flowchart representing the top-level execution methodology of the Solution proposed

A. Phase – 1 DV Flow for legacy features using Silicon Proven Design

The Reverse Connectivity map is generated from silicon proven design after compiling and elaborating the design in Jasper Gold tool. When the checkers are back proven on same design, all checks should pass. This connectivity map can be treated as a golden map which has all the connectivity information of legacy design. This reverse connectivity map can be further imported to newer spin-off designs in a formal environment where the checkers are proven to identify whether the legacy features are intact. The failures may be due to a valid RTL bug which can be reported and get it fixed. Some checkers might be no longer valid in spin-off device due to port naming changes or cut-down features, will also result in failures which can be reviewed and checkers can be modified. In either case we rerun the setup until the there are no more failures. However, we can live with some expected failures due to cut-down features in spin-off devices and these should remain intact across the flows. The final Legacy reverse connectivity map, is now ready to export to next incremental releases of SoC. Fig. 3 represents the overall flow for execution of phase -1 DV.

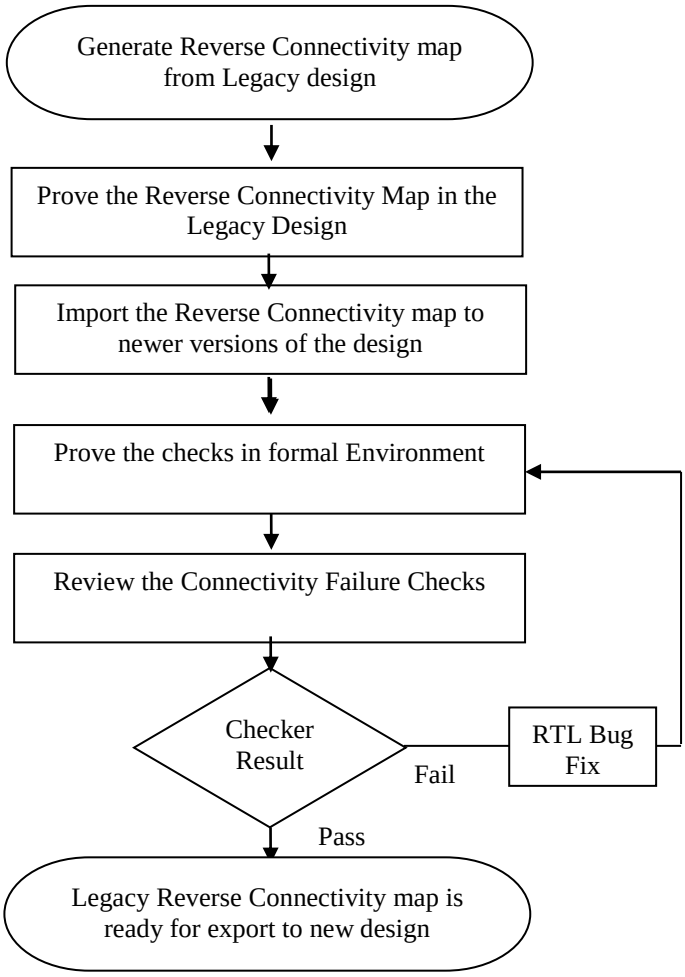

Figure 3. Flowchart representing the Phase-1 of the DV flow methodology for proving the legacy features using Silicon Proven Design

B. Phase – 2 DV Flow for Incremental releases of SoC

The legacy map can be now exported to new Incremental SoC releases. The failures seen in the previous phase should be intact, any new failures will impact the legacy connectivity and hence new failures needs to be debugged and get the RTL fixed. The setup needs to be rerun until the failures from previous phase are intact.

Also, a new connectivity map needs to be generated from first release of the spin-off device. This connectivity map will be used to prove the connectivity of all further incremental releases of SoC ensuring that all the new feature set connectivity is intact. The flow needs to be re run until there are no failures .

Hence for every incremental release of SoC , the legacy features as well as new features connectivity from previous SoC release will be ensured to be correct.

This phase needs to be rerun until a stable and bug free SoC release is available. Fig. 4 represents the flowchart for DV flow of phase-2

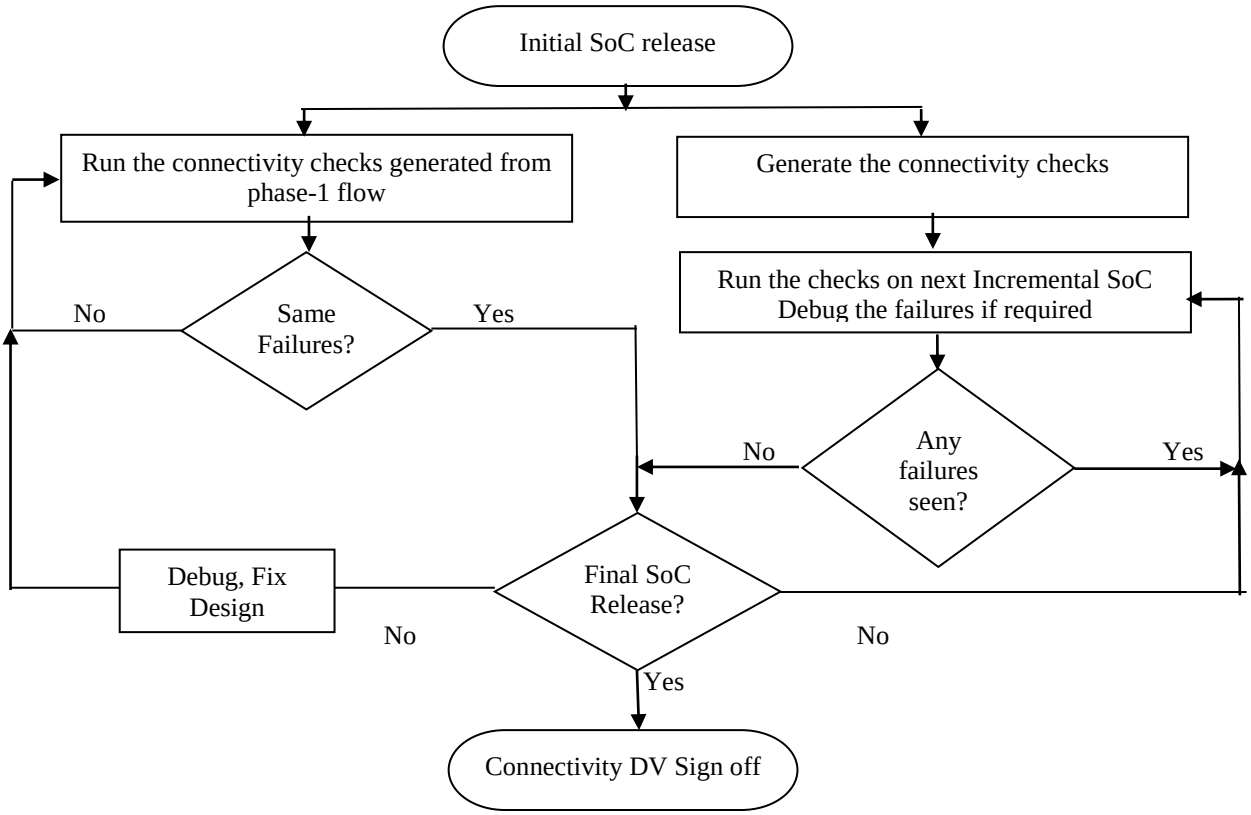

Figure 4. Flowchart representing Phase-2 of DV flow for proving the connectivity for incremental changes in SoC

V. FASTER EXECUTION CAPABILITY

The reverse connectivity based solution helps in faster feedback of the connectivity changes and hence faster bug detection capability without going to actual simulation environment. This solution is independent of the area of the device. Fig.5 shows the chart of execution time comparison across multiple devices in a platform. The overall simulation effort increases with increase in number of releases. The Formal based Reverse Connectivity approach enables faster execution with no compromise in quality. Given its impact on execution time this flow helps in long way across the road map of the devices.

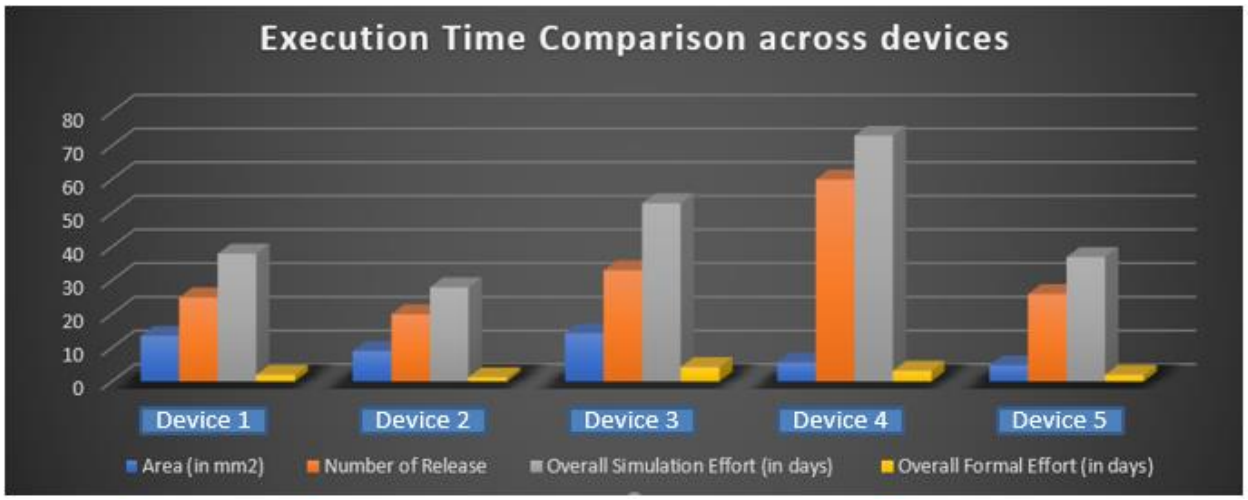

Figure 5. Graph representing the comparison of execution time between overall simulation effort vs. overall Formal effort across devices in a platform with different areas of device

VI. CRITICAL DESIGN ISSUES FOUND FROM THIS FLOW

Out of many connectivity change findings through this flow, some of the generic critical findings which would have led to increase in design cycle time if not found through this flow were described below

A. Missing power isolation between power domains

If there is no isolation between source and destination domains, then entire destination domain gets corrupted if source domain is in low power mode. This issue can be found in low power simulations. Effort required to find this issue is Bring up of testbench which takes 2 days and regression run time which takes 1 day. This issue can be caught through the reverse connectivity approach without actually entering into the simulation environment after a SoC release.

B. Missing reset Connection

Since reset connection is missing, SoC will never lifted out of reset and hence power up is not possible and wake up from low power mode also gets impacted. This issue can be found through testbench bring up which takes approximately 6 hours.

C. Missing Pad Connection with peripherals

Wakeup from low power mode gets impacted if wakeup source is an external trigger. Bug detection time is 6 for testbench bring up.

Fig.6 gives the overall debug turn around time which includes the environment updates, testbench bring up time, regression run time, failure debug and RCA. Formal on other hand is faster to setup and faster to detect bugs.

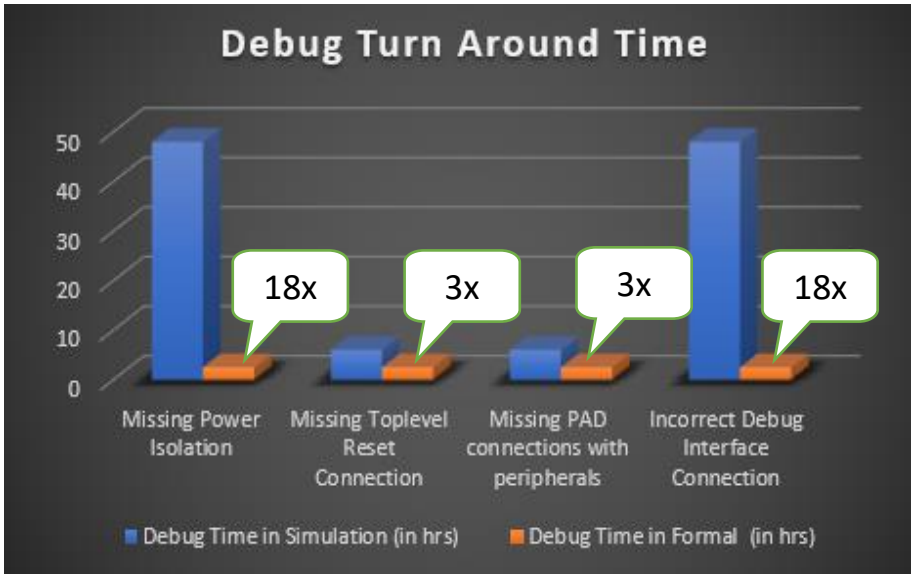

Figure 6. Graph representing the debug turnaround time when issues were found through Simulation approach vs. the Formal solution proposed

VII. CONCLUSION

Reverse Connectivity solution is helpful to bridge the gap between SoC Integration and DV Environment for faster feedback on connectivity changes or misses. This solution can identify the integration changes between spin-off and Legacy devices and ensure that all the legacy features were intact without going to actual simulation environment. This approach also helps in faster understanding of the design changes or bug fixes between the incremental SoC releases. As a next step, flow can be automated for achieving the plug and play kind of approach

REFERENCES

[1] A. Nikolov, E. Pencheva, I. Atanasov, published 6 June 2016, Computer Science, 2016 IEEE International Black Sea Conference on Communications and Networking, Formal Verification of connectivity management models in M2M communications

[2] Pedram Riahi – Raytheon, User2User North America 2020, Achieving 100% Code Coverage: What Used to Be Hard Is Now Easy with an Automated, Exhaustive Flow

[3] Sai Rama Krishna Nalla, Arsen Manukyan, Udupi Harisharan, User2User North America 2020, Exhaustive Verification of SoC Interconnect Including Conditional and Variable Delay

[4] Ping Yeung, Mark Handover, Abdel Ayari, User2User North America 2020, Formal Verification Experiences: Silicon Bug Hunt with “Deep Sea Fishing”