Novel approach for SoC pipeline latency and connectivity verification using Formal

Deepak Mohan, *Senior Engineer, Qualcomm*, Senthilnath Subbarayan, *Senior Staff Engineer, Qualcomm*, Sandeep Kumar, *Principal Engineer, Qualcomm*

I. INTRODUCTION

In a SoC (System on Chip), various design blocks communicate each other with different protocols say AHB (Advanced High-performance Bus), AXI (Advanced eXtensible Interface) etc. and pipelines are the design elements introduced between these blocks to delay the entire protocol to meet the timing requirements based on physical design feedback, as shown in Fig. 1. Thus, each pipeline will have a specific latency and an FSM (Finite State Machine) based on a particular protocol within it. The latency verification and sideband signal coverage for these pipelines in SoC is usually done during late stages of functional verification, say performance verification. Since these occur late in the cycle, a flow has been proposed to verify these latencies much earlier in front end verification using Formal approach. This paper explains a way to verify the correctness of pipelines/bridge latency and protocol compliance between blocks using formal connectivity app through an Automated Pipeline Utility Engine developed using Python. This left shifts the verification cycle by enabling pipeline/bridge latency and connectivity verification well ahead in the flow. The verification for pipeline latency contains connectivity checkers which implicitly verifies the connectivity between two modules. The intent is to generate CSV (comma-separated values) files through Automated Pipeline Utility Engine using pipeline and connectivity specification, thereby making the entire setup automated.

II. RELATED WORK

The main challenge in pipeline verification is capturing the large number of protocol-based connections between various blocks and feeding the input to CCAPP (Connectivity Checking Application) in the required format. Also, to check a single connectivity, the enable condition required will depend on other signals of same pipeline, which again varies from protocol to protocol. The setup time required to start the pipeline verification (which includes the effort to create CSV files for all the connections of all pipelines at SoC, identify the clock and reset of the pipeline, identify the enable conditions to check a connectivity and identify any general constraints which can be applied to ensure connectivity), typically takes around ~3 weeks of effort considering 100-200 pipelines in a design. Any change in pipeline design during the tenure of the project cycle, retriggers pipeline verification. To minimize this time window is the key challenge in SoC pipeline verification automation.

Figure 1. SOC pipeline connectivity

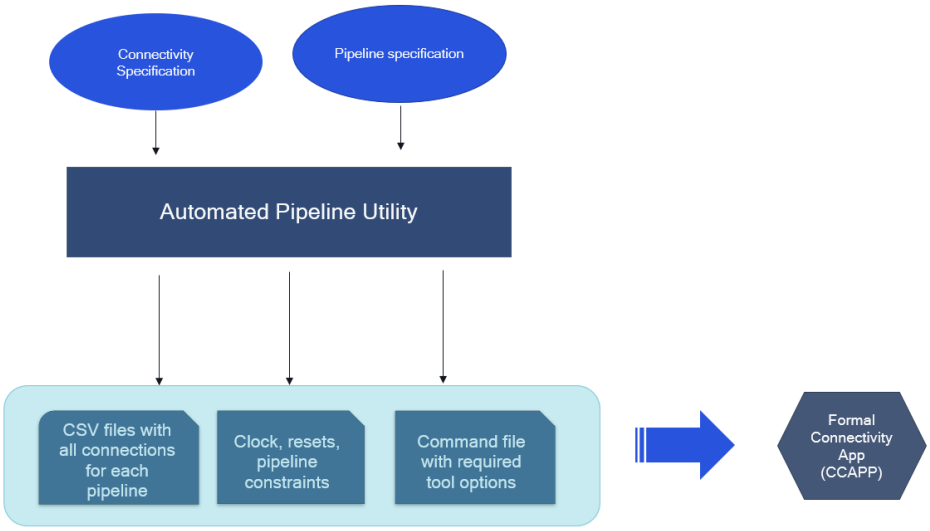

Figure 2. Novel pipeline verification flow

III. APPLICATION

The Automated Pipeline Utility Engine consumes connectivity specification and pipeline specification as input and generates CSV files, clocks, resets, and the constraints required for each pipeline, along with the settings in the required format for CCAPP. The pipeline connections at SoC can be of any protocol [say AXI, AMBA (Advanced Microcontroller Bus Architecture) protocols like ACElite (AXI Coherency Extension), CHI (Coherent Hub Interface)], which are also taken care as part of this automation through formal assertions. With this approach, we verify the connectivity between two modules along with open connections (as shown in Fig. 3) and signals which are not passing through the pipelines (as shown in Fig. 4).

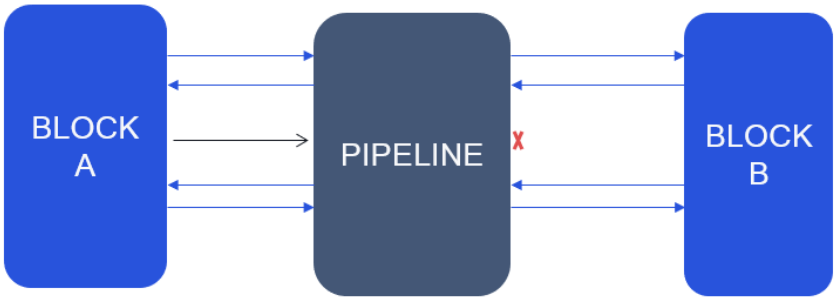

Figure 3. Open ports in pipeline connection

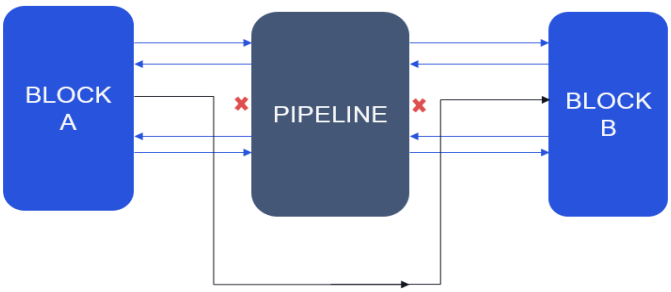

Figure 4. Connections not going through pipeline

IV. RESULTS

New Approach helps and enables SoC DV (Design Verification) engineers to catch issues which are hard to find using functional verification and automatically generates the input connectivity files required for SoC pipeline verification to close the latency and connectivity checks well ahead in verification cycle. The key advantages of this approach are listed in Fig. 5. Further the automation saves weeks of effort spent on pipeline verification setup alone and eliminates the redundant verification effort on design changes, as shown in Fig. 6. This flow has been deployed and proved in multiple projects, thereby enabling a standard approach for pipeline verification.

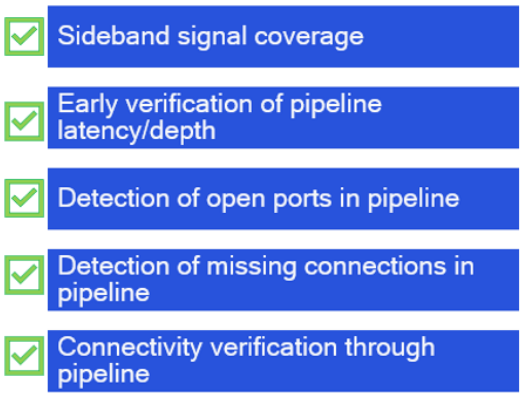

Figure 5. Key advantages

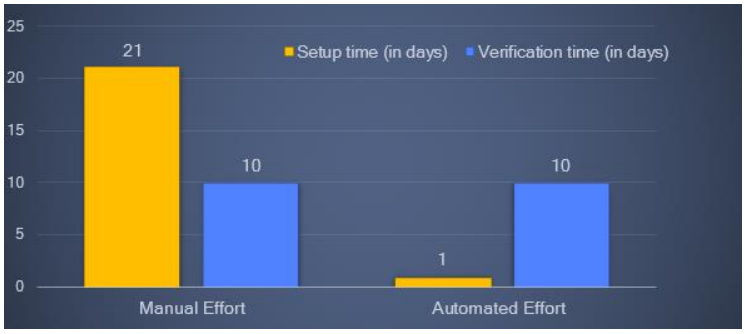

Figure 6. Effort saving comparison

V. CONCLUSION

With this approach, entire SoC pipeline latency and connectivity is verified through formal connectivity checks in SoC DV, resulting in left shift of verification cycle. Also, the enabled automation in pipeline verification reduces the environment setup time by generating the required input files for verification with a single click deployment. This paper concludes the usage of formal solution at SoC pipeline verification which reduces the time/effort thereby increasing the quality of verification.

ACKNOWLEDGMENT

The authors would like to thank entire formal team for useful discussions and suggestions. Also, the support from entire management is acknowledged.

REFERENCES

[1] VC Formal Verification User Guide, Version Q-2020.03, March 2020