Framework for Automated Connectivity Checks for core and SOCs

Avinaba Tapadar, Akash Singh, Mohit Solanki, Raghu B R, Srobona Mitra
{atapadar, akassing, mohisola, rbr, srobmitr}@qti.qualcomm.com
Qualcomm India Private Limited

***Abstract*- Connectivity verification is a time-consuming but critical task in the realm of verification of modern System-on-Chip (SoC) designs. The sheer volume of connections, often reaching hundreds of thousands, poses a significant challenge. Manual construction of the connectivity sheets is not only troublesome, but also comes with high probability of bugs creeping in the design. To streamline this process, our work focuses on left shifting the connectivity verification step toward earlier stage of design cycle. We employ formal verification techniques to find out connectivity mismatches between the specification and the implementation, which helps us to detect critical bugs. We also propose the syntax and semantics for carrying out rule-based static checks for verifying SOC integration connectivity.**

Our tool has been adopted by multiple teams and applied to various projects, instilling confidence in SoC designs.

I. Introduction

Connectivity verification [1] stands as one of the most time-consuming and critical tasks in the verification of System-on-Chip (SoC) designs [2], primarily due to the sheer volume of checks required and the severe repercussions of any bugs. At the SoC level, the number of connections often reaches into the hundreds of thousands, if not more. Traditionally, connectivity information is manually filled in specification releases. Errors in these specifications are frequently detected at the later stage of design cycle, where it becomes challenging to ascertain whether the bug originates from the specification or the RTL design. Debugging at this stage necessitates the involvement of multiple teams, including Core Design, SoC Design, and SoC Design Verification (DV).

Our work aims to streamline the prognosis step by formally verifying the mechanism for different applications, which includes various domains including memory redundancy, interrupt control, fuse configuration, complex power reduction mechanisms, etc. at the earlier stage of core level design cycle. We utilize the Connectivity app from EDA vendor formal tools to facilitate this process. Leveraging the Formal Connectivity app, we've successfully detected critical bugs, enhancing overall verification quality. Additionally, we introduce a syntax and semantics framework for conducting rule-based static checks to verify SoC integration connectivity.

Our tool has been adopted by multiple teams and applied to several projects. It has proven its effectiveness by identifying numerous bugs, thereby enhancing the quality of verification and instilling greater confidence in SoC designs.

II. Application

Our developed tool facilitates connectivity verifications at both the core and SOC levels. Below is an overview of our contributions:

A. Core Level Formal Verification

a. Formal Connectivity Verification: We deploy formal connectivity checks to ensure accurate connectivity verification within minutes.

b. Our tool facilitates efficient checks for different applications, which includes various domains including memory redundancy, interrupt control, fuse configuration, complex power reduction mechanisms, etc.

B. SOC Level Static Checks

a. Syntax and Semantics for Connectivity Rules: We define the syntax and semantics for specifying connectivity rules at the SOC level.
b. Inter-Core Connectivity Checks: Our tool examines SOC-level inter-core connectivity based on user-provided rules.
c. Automated Static Checks: We perform automated static checks on SOC-level connectivity.
d. Connectivity Coverage Metric: Our tool provides a metric to assess SOC-level connectivity coverage.

In our pursuit to simplify connectivity verification for System-on-Chip (SoC) designs, we've crafted a robust syntax and semantics for specifying SOC connectivity rules. Notably, we've introduced two types of wildcards: "*" to denote all matching cases and "+" to represent at least one matching case. By leveraging these rules, we enhance the accuracy and efficiency of connectivity checks across SoC components.

Following are a few examples:

TABLE I
RULE BASED STATIC CHECKS

Src instance	Src port	Dest instance	Dest port	Semantics
u_noc	**a_irq**	**u_modem**	**irq_in**	***u_noc.a_irq* → *u_modem.irq_in***
u_noc	**a_irq**	**u_modem**		***u_noc.a_irq* → any one port in *u_modem***
u_noc		**u_modem**	**irq_in**	**any one port from *u_noc* → *u_modem.irq_in***
u_gcc	**gcc_apc_clk**	**u_modem**	***_clk**	***u_gcc.gcc_apc_clk* → all ports matching with *u_modem.*_clk***
u_gcc	**gcc_apc_clk**	**u_modem**	**+_clk**	***u_gcc.gcc_apc_clk* → at least one port matching with *u_modem.*_clk***
u_gcc	**+_clk**	**u_***	**+_clk**	**At least one *clk* signal from *u_gcc* → at least one *clk* signal in all *u_** modules**

There is also provision to provide multiple destinations for a source.

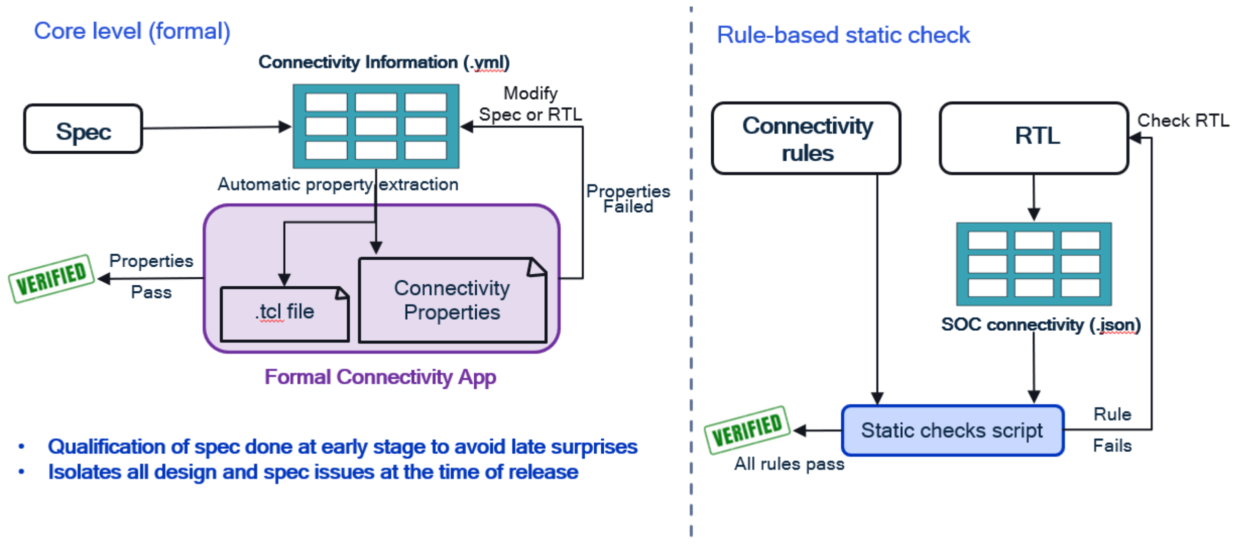

Figure 1. Connectivity Checking framework for SoC and Subsystem

III. Results

Following are some of the bugs which we caught during the initial deployment phase:

A. Project 1: at least 6 bugs were caught
B. Project 2: >50 bugs caught at a very early stage of the design
C. Project 3:
    a. Core level: 9 bugs caught at core-level
    b. SOC level: Bugs caught at SOC integration level before DV and patches were provided.

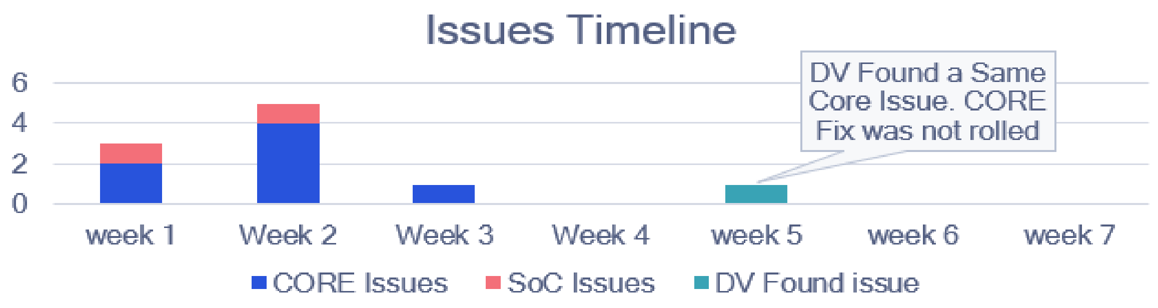

Figure 2. Timeline vs Issues caught

IV. Conclusion

In our pursuit of enhancing System-on-Chip (SoC) design quality, we have devised an innovative approach that significantly impacts the verification process. By shifting the verification step to an early stage, we pre-emptively address bugs that might otherwise halt development later. The complexity of multiple variables is mitigated through formal checks across different cores and static checks within the SoC.

Our end-to-end automated methodology is a game-changer. It slashes verification time from multiple days to just 30 minutes at the core level and approximately 1 hour at the SoC level. Furthermore, this efficiency boost is achieved while enhancing the ease and quality of the verification. Our proposed approach has yielded promising results across various projects, catching critical bugs, and instilling confidence in the design.

Notably, our solution seamlessly integrates into the spec-release system, providing a one-click solution. It automatically detects mismatches between the RTL and the specification during every spec release. As we forge ahead, this streamlined connectivity validation methodology continues to empower SoC development, ensuring robustness and accelerating progress.

Acknowledgment

Multiple teams within Qualcomm (project DV, methodology) have worked in the development/validation of the flows.

References

[1] S. K. Roy, "Top Level SOC Interconnectivity Verification Using Formal Techniques," *2007 Eighth International Workshop on Microprocessor Test and Verification*, Austin, TX, USA, 2007, pp. 63-70, doi: 10.1109/MTV.2007.22.

[2] H. Saafan, M. W. El-Kharashi and A. Salem, "SoC connectivity specification extraction using incomplete RTL design: An approach for Formal connectivity Verification," 2016 11th International Design & Test Symposium (IDT), Hammamet, Tunisia, 2016, pp. 110-114, doi: 10.1109/IDT.2016.7843024.