2025 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION INDIA

Novel Formal Equivalence Approach to Verify Scalable Architecture in GPU

Sanjana Jain, Ipshita Tripathi
Qualcomm India Pvt. Ltd

***Abstract* -To scale up and down performance based on program requirements, GPU architecture is composed of both scalable and unscalable design blocks to enable flexible performance tuning through up-scaling or down-scaling. Scalable blocks can be instantiated multiple times, allowing a wide range of configurations by selectively disabling certain instances. However, this flexibility introduces significant verification challenges.**
With traditional verification, disabling different combinations of components results in an explosion of test cases, due to the large number of scalable blocks and their possible configurations, which slows down the process and consumes excessive resources. Moreover, disabling different instances of identical blocks can expose unique corner-case behaviors, complicating the verification process further.
To address these challenges, we propose a Sequential Equivalence Checking (SEC)-based formal verification approach. Our approach significantly reduces the verification effort and reruns, complementing simulation techniques. By verifying the complete design—including faulty components identified for disabling—our method ensures robust coverage and reliable signoff for complex GPU architectures.

I. INTRODUCTION

GPU performance requirements change from program to program depending on the ever-shifting customer landscape and requirements. To cater to a wide spectrum of requirements, scalable GPU architecture is used to be able to reduce time to market and provide flexibility. To achieve this, GPU Architecture consists of blocks which are divided into "scalable" and "unscalable" design types for ease of upscaling or down-scaling based on performance requirements of program. Wide variety of configurations can be achieved by removing different scalable blocks. To ease this up/down scaling, features were added in GPU architecture which ensured that if we remove one scalable block vs another identical scalable block the performance and functionality should remain intact.

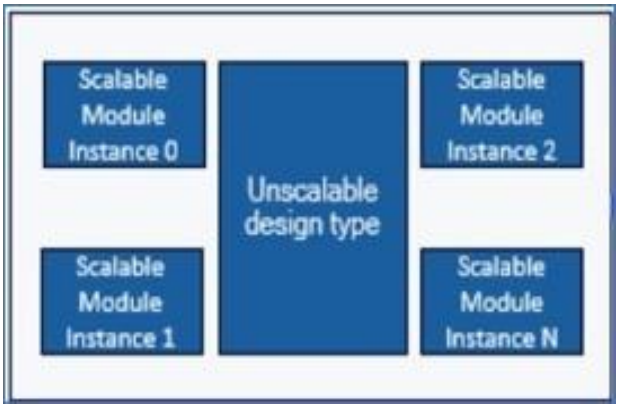


Figure 1. Scalable N module instance architecture of GPU

This added tremendous verification complexity. There were many scalable blocks with multiple instances lead to many combinations of disabled components. Using traditional methods of verification, many combinations of disabled components from the design were possible, which resulted in an exponential increase in the number of tests that needed to be run. This was a difficult challenge given ever shrinking project timelines, ever-increasing design complexity and limited resources. Disabling one instance vs another often exposed new corner case scenarios which were previously unseen in another configuration and thus exhaustive verification was required to be able to catch all issues and ensure quality.

Verification of such complicated GPU systems frequently encounter bottlenecks in discovering all corner-case design problems. Thus, in order to verify such designs, we required a unique solution. Traditional verification approach led to the addition of large number of tests. To reduce these re-runs across multiple configurations, we proposed a Sequential Equivalence approach (SEC), which will eventually lower the overall verification effort and enable us to sign off on verification using a formal based verification strategy in conjunction with traditional simulation-based approaches.

II. DESIGN OVERVIEW

GPU Architecture consists of blocks which are divided into "scalable" and "unscalable" design types for ease of upscaling or down-scaling based on performance requirements of program. Scalable blocks refer to those design elements which can be scaled up and down seamlessly by addition or removal of multiple instances of the block. Unscalable blocks refer to those design elements which do not scale seamlessly – often referring to blocks which broadcast traffic to multiple scalable blocks or aggregate traffic from multiple scalable blocks.

In our design, only one faulty component can be disabled from N components at a time. We worked on downsizing the scalable architecture by disabling one instance of a scalable module. An identifier identifies which component is to be disabled and is to be disabled. The design we are verifying is the entire design, including the faulty components.

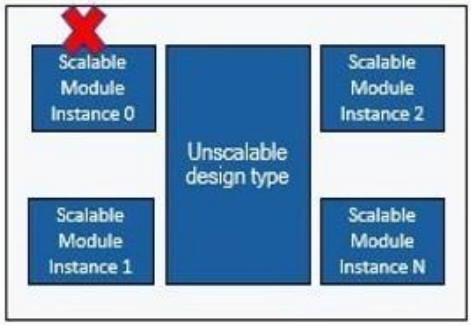


Figure 2. Module Instance 0 is disabled enabling downscaling of performance

III. CHALLENGES FACED IN TRADITIONAL VERIFICATION

The removal of instances to downscale a design added complexity to simulation-based flows. Large number of new testcases were introduced which would increase the number of scenarios that needed to be covered to exhaustively verify this feature of the design.

Formal property-based flows for such a large design would not yield any productive result as it would be overkill for the problem at hand.

Sequential Equivalence Checking is the process of proving formal equivalence between two non-state matching implementations of a given design specification. The design was such that out-of-the-box sequential flows will also not help as there were requirements from an input point of view which would need to be honored for the designs to behave equivalently.

Thus, a custom Formal Equivalence based novel approach was developed to be able to verify this feature. This approach would not guarantee correctness, but given a correctly working configuration, it would guarantee that other configurations behaved equivalently.

In the following sections we attempt to explain the methodology in greater detail.

IV. PROPOSED METHODOLOGY

GPU Architecture consists of scalable and Unscalable blocks. Scalable blocks refer to those design elements which can be scaled up and down seamlessly by addition or removal of multiple instances of the block. Unscalable

blocks refer to those design elements which do not scale seamlessly – often referring to blocks which broadcast traffic to multiple scalable blocks or aggregate traffic from multiple scalable blocks.

In the methodology that we talked about today, we discussed the methodology with respect to downsizing the block by disabling one instance of a scalable block. We want to ensure that irrespective of which instance we disable, the performance and functionality remain intact. The methodology relies on being provided the identifier of which instance is disabled. It also works in complementary fashion to the simulation efforts in verifying the block as described below.

A Formal Equivalence based novel approach has been developed for proving equivalence of two different disabled components and comparing whether the outputs are equivalent irrespective of which identical component is disabled. For example, if there are 3 identical components, irrespective of which of the three has a defect, the overall design remains unchanged to achieve a seamless scale up and down.

Formal equivalence is generally never used to ensure correctness of a system. It is typically used in cases where correctness is established, either through previously proven silicon or previously tested design. Equivalence is then proven against a different version of this design to ensure multiple configurations are tested.

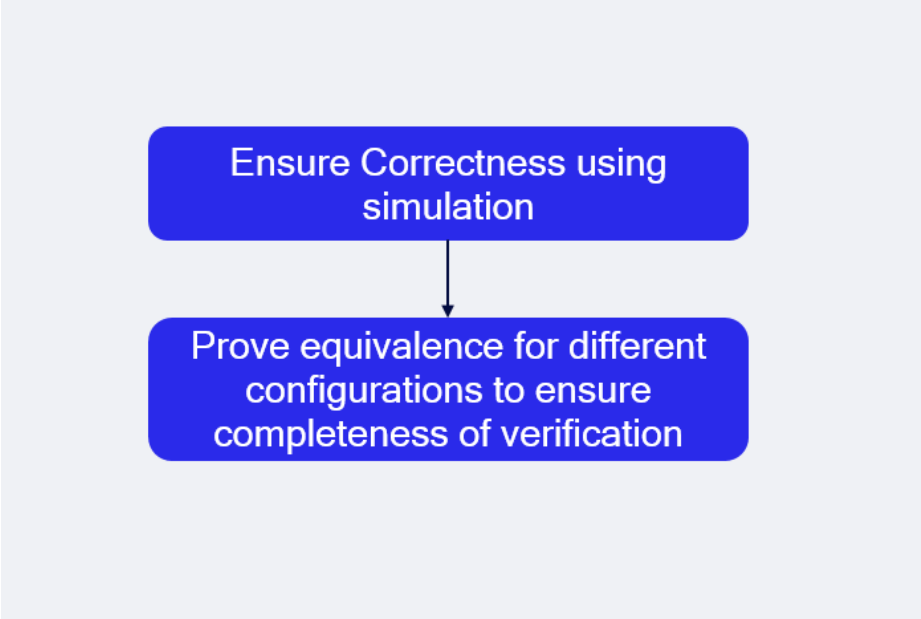


Figure 3. Steps for completeness of verification

In our proposed methodology, we want to ensure that given that one configuration has been tested using simulation, is there any way to ensure other configurations are equivalent.

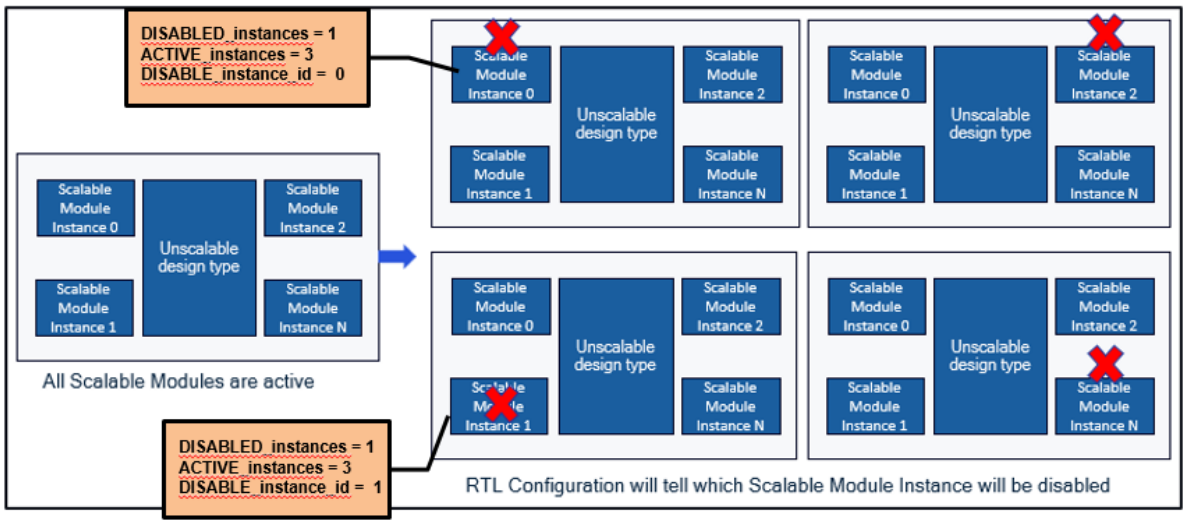


Figure 4. Multiple configurations possible due to 1 instance being disabled

As seen in the figure 4 above, there are multiple configurations possible by disabling one instance of a scalable block. Using our methodology, we want to ensure that suppose we disable instance 2 versus instance 3 we get the same functionality and performance.

In order to achieve this, we have divided the GPU design into multiple smaller blocks, around total 40 of them total and we used customized flow to be able to retrofit our design to the traditional sequential equivalence flow.

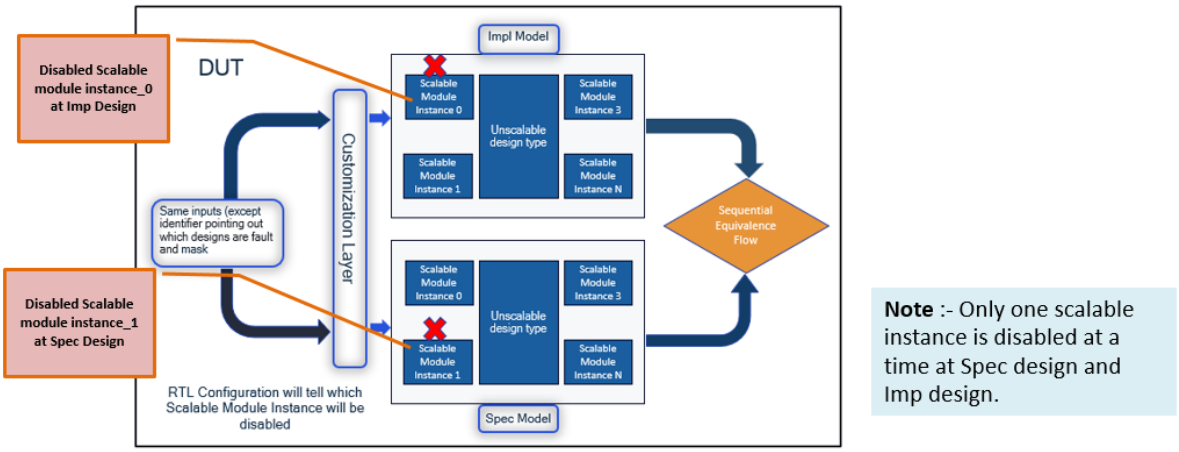
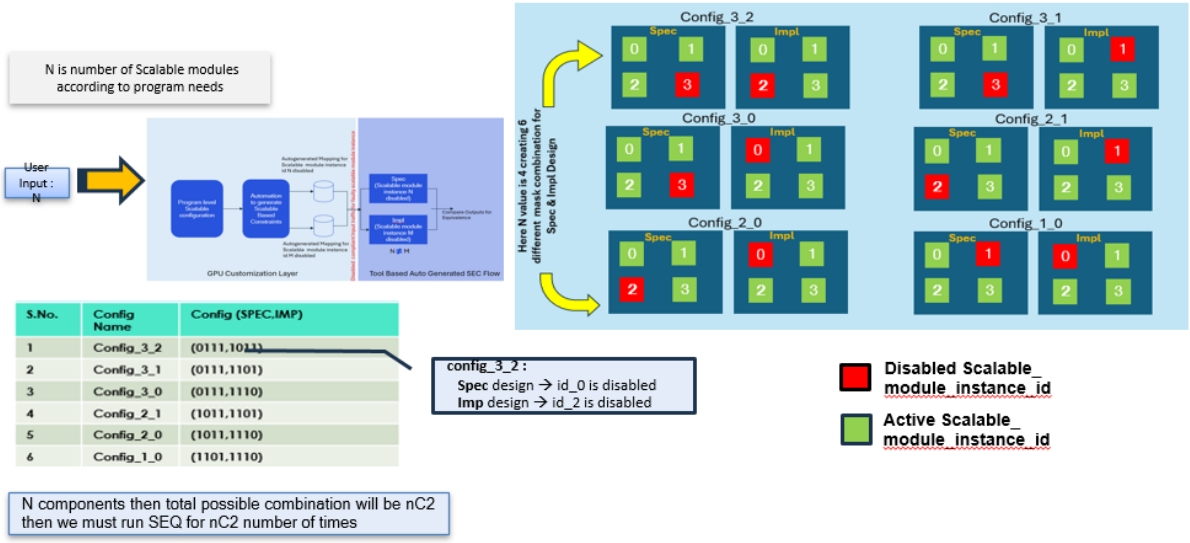


Figure 5. Proposed Flow using formal custamizable layer

The creation of the customizable layer is central to be able to use the traditional sequential equivalence flow. Our designs are such that if a given instance is disabled, it will expect no traffic to flow to the disabled instance. Also the traffic needs to be correctly routed depending on which instance is disabled. Let us explain with an example.



S.No.	Config Name	Config (SPEC,IMP)
1	Config_3_2	(0111,1011)
2	Config_3_1	(0111,1101)
3	Config_3_0	(0111,1110)
4	Config_2_1	(1011,1101)
5	Config_2_0	(1011,1110)
6	Config_1_0	(1101,1110)

Figure 6. Various configurations disabled between two equivalence problems

For example, between two configurations, say 3 disabled in one configuration and 2 disabled in another, the traffic is stopped to 3 in the "spec" or first version of the design and stopped to instance 2 in the "impl" or the second version of the design. It is also important to ensure that the traffic of instance 2 of spec is routed to instance 3 of impl for the configurations to behave equivalently. Thus to ensure that these configurations are considered equivalent, we added a customization layer as shown below which automated the abpove behavior depending on the configuration that was being tested.

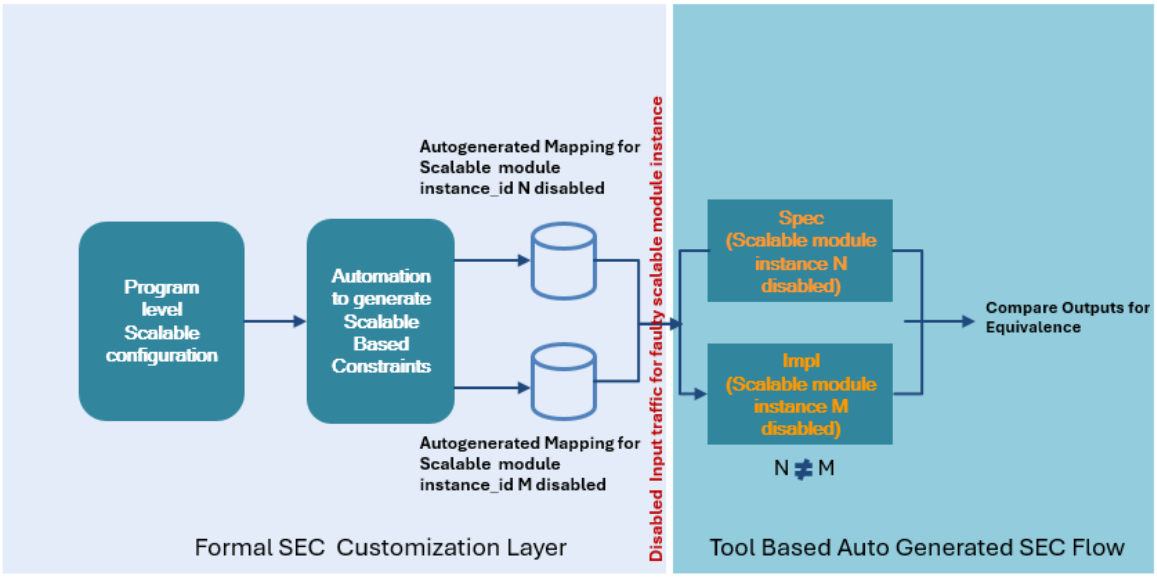


Figure 7. Formal SEC Custamizable Layer

IV. RESULTS, CONCLUSION AND FUTURE WORK

Using this methodology, we were able to identify 7 RTL bugs and 5 spec bugs. We were able to achieve 100% convergence for 75% of GPU blocks which were of scalable type. For the unscalable blocks we saw good success in bug-hunting mode, but full convergence was not seen and we are working with tools to achieve this in upcoming days. We also achieved a 60% reduction in the re-runs on future projects once the high effort first project was completed.

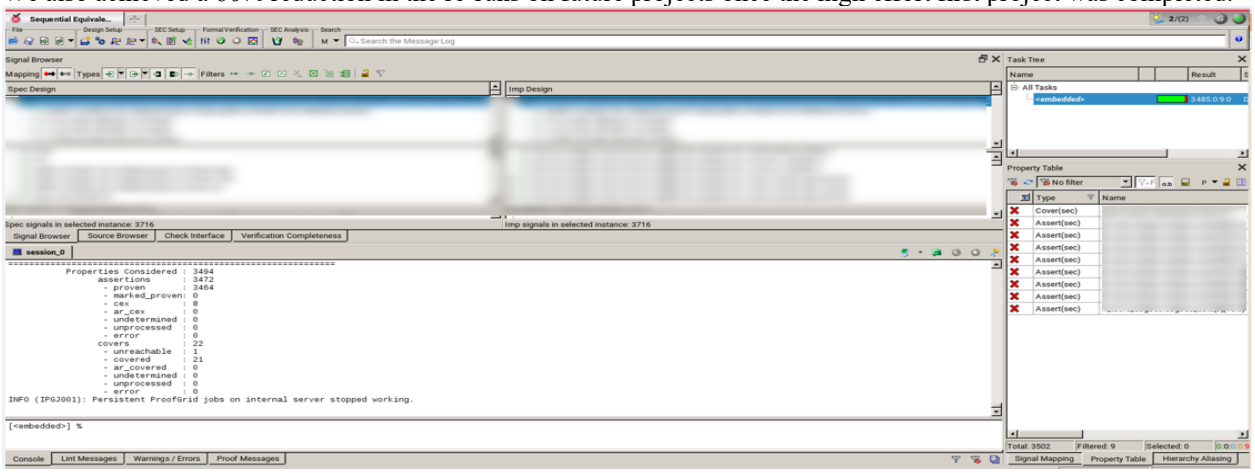


Figure 7. Converged Scalable Block

We plan to extend this methodology in upcoming days to encompass more than one instance being disabled.

REFERENCES

[1] You Li, Guannan Zhao, Yunqi He, Hai Zhou, "DE2: SAT-Based Sequential Logic Decryption with a Functional Description", *2025 Design, Automation & Test in Europe Conference (DATE)*, pp.1-7, 2025.

[2] Kuehlmann, A., van Eijk, C.A.J. (2002). Combinational and Sequential Equivalence Checking. In: Hassoun, S., Sasao, T. (eds) Logic Synthesis and Verification. The Springer International Series in Engineering and Computer Science, vol 654. Springer, Boston, MA. https://doi.org/10.1007/978-1-4615-0817-5_13

[3] Baumgartner, Jason, et al. "Scalable sequential equivalence checking across arbitrary design transformations." *2006 International Conference on Computer Design*. IEEE, 2006.

[4] Mneimneh, Maher N., and Karem A. Sakallah. "Principles of sequential-equivalence verification." *IEEE Design & Test of Computers* 22.3 (2005): 248-257.

[5] Mathur, Anmol, et al. "Sequential equivalence checking." *19th International Conference on VLSI Design held jointly with 5th International Conference on Embedded Systems Design (VLSID'06)*. IEEE Computer Society, 2006.