2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION INDIA

Enhancing Arbitration Integrity: Formal Verification of Weighted Round Robin Arbiter in High-Performance Graphics

Kolluri Ajay Kumar, Bagadi Usha Rani, Verma Vichal, Choradia Mohit.
Intel Corporation
ajay.kumar.kolluri@intel.com; usha.rani.bagadi@intel.com;
vichal.verma@intel.com; mohit.choradia@intel.com.

Abstract- In the realm of high-performance graphics computing, the orchestration of shared resources among a multitude of clients is a critical challenge that necessitates robust and sophisticated arbitration mechanisms. The Weighted Round Robin (WRR) arbiter is particularly instrumental in such environments, where the equitable distribution of bandwidth is crucial for rendering tasks and real-time graphics processing. Its weighted bandwidth allocation strategy is adept at balancing fairness with performance, a requirement that is paramount in graphics systems where computational workloads and memory access patterns can vary significantly. This study presents a detailed formal property verification of a WRR arbiter, which has been meticulously engineered with advanced features to meet the exacting demands of high-throughput graphics computing architectures. The verification process rigorously examines the arbiter's functionality to ensure it adheres to its design specifications, confirming its ability to efficiently prioritize and allocate resources amidst the complex interplay of multi-client graphics processing scenarios. The findings reinforce the WRR arbiter's suitability for high computing graphics, where its role is critical in maintaining system performance and ensuring the fair treatment of diverse graphics workloads.

I. Introduction

This paper examines the design and formal verification of a Weighted Round Robin (WRR) arbiter, a key component in bandwidth management for high-performance computing. The arbiter intelligently handles various request types and maintains system stability through a flexible 2-D array of weights, client prioritization, and advanced features like back-to-back data streaming. The design operates effectively in both standalone and multilevel modes, ensuring efficient conflict resolution and resource allocation.

The verification of this complex arbiter posed significant challenges, addressed through a comprehensive formal analysis that confirmed the arbiter's adherence to its specifications and operational efficiency. The process uncovered and resolved around 50 critical bugs, underscoring the importance of formal verification in maintaining the integrity and performance of such intricate systems. The resulting FPV testbench is a robust tool for future development, ensuring the arbiter's reliability in forthcoming high-throughput computing environments.

II. Design overview

The arbiter's design under test incorporated a 2-D array of weights to ensure stable bandwidth allocation during operation, with the flexibility to adjust weights only during idle periods. It adeptly handles a variety of request types, including read, write, and read return requests, while enforcing command and data credit constraints to prevent resource over-commitment. The arbiter supported back-to-back data streaming for efficient multi-cacheline transactions to optimize command bus utilization. Arbiter is used in standalone mode (figure 1) and multilevel arbitration mode, for source and destination conflict resolution with grant from second level looped back to input of first level arbitration indicating the requesting client is serviced (figure2). Furthermore, the arbiter implements client stalling to manage busy destinations and employs lane classification to prioritize clients, with diamond lane clients receiving guaranteed access for a set number of cycles. The formal verification process presented in this paper rigorously examines the arbiter's functionality, ensuring that it adheres to its specifications and correctly prioritizes and grants access to clients under various scenarios. The exhaustive verification using FV confirms the arbiter's

ability to manage complex interactions and maintain system performance and fairness, making it a suitable choice for high-performance computing environments with stringent resource management requirements.

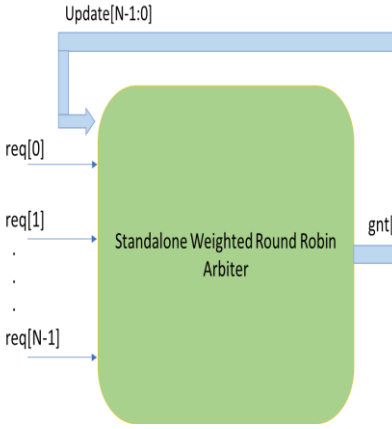

Fig 1 Standalone WRR Arbiter

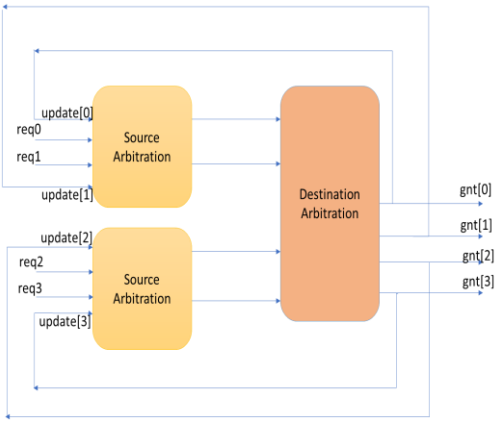

Fig 2 Multilevel arbitration for source and destination conflict resolution.

III VERIFICATION CHALLENGES

Verification of a Weighted Round Robin (WRR) arbiter presents a set of unique challenges due to the complexity of its features and the need to ensure fairness, optimality, and efficiency in resource allocation among multiple clients. This paper addresses the verification of a WRR arbiter designed to manage bandwidth allocation through a 2-D array of weights, handle various request types, and incorporate advanced features such as command/data credit allocation, back-to-back data streaming, meta-packing, client stalling, and lane classification.

The verification process aims to address several critical aspects to ensure the arbiter's correct functionality.

TABLE I
CLASSIFICATION OF CHECKERS BASED ON FEATURES

Checks/Feature	Expectation
Progress and Starvation	The verification must prove that every client makes progress and is not subjected to starvation once its request is asserted, given the availability of all resources.
Optimality of Operation	The arbiter must be formally verified to ensure that a grant is issued in any cycle where at least one request is not blocked by resource constraints, thereby proving the optimality of operation.
Zero Weight Grant Prevention	It must be verified that clients with zero weight are never granted access and that no client is granted access more times back-to-back than its weight allows, especially if another client is asserting a request during those cycles.
WRR Selection Property	The verification must confirm the WRR selection property by calculating client priorities based on weights, requests, credit availability, and stall conditions

Diamond Lane Client Grants	The arbiter must be proven to grant access to at least one diamond lane client up to the programmed value of diamond lane timeout counter whenever any diamond lane client is requesting.
Non-Diamond Lane Client Grants	Verification must ensure that a non-diamond lane client is granted access within a worst-case window of diamond lane timeout counter +1 grants if any non-diamond lane client is requesting.
Write Request Prioritization	The arbiter must prioritize write requests over read requests to improve data bus utilization, and this prioritization must be formally proven.
Multi-Cacheline Transaction Exclusivity	The verification must prove that for any multi-cacheline write or read return request granted, no other write/read-return request is granted until the transaction is complete. However, zero-length write requests must be allowed grants during this window.

The formal verification of these properties is essential to ensure that the WRR arbiter operates as intended, providing fair and efficient access to shared resources while optimizing system performance. The verification challenges outlined in this paper are addressed through rigorous formal analysis, ensuring that the arbiter adheres to its design specifications and meets the high standards required for complex, high-throughput computing systems.

IV. Verification Strategy

Within the scope of design verification, we concentrated solely on the Formal Property Verification (FPV) environment tailored for this specific functional unit block. To thoroughly validate the arbiter's features, we adopted a layered decomposition strategy, progressively unveiling and verifying layers of functionality. Initially, we imposed over-constraints on the design, such as disabling Diamond Lane requests and multi-cacheline writes. This simplification allowed us to concentrate on verifying fundamental aspects like weight balancing and the prioritization of write requests over read requests. As verification progressed, these constraints were gradually lifted to examine the full spectrum of the arbiter's capabilities.

Subsequently, we incorporated forward progress checks for each client. These checks ensured that a client with sufficient credits and no stall condition would receive a grant within a calculable number of cycles, defined by the product of the number of requesters, the maximum weight, and the maximum data length. To uphold fairness, we implemented checks that guaranteed a client, once granted access, would continue to receive grants for a number of cycles at least equal to its weight, provided its request remained asserted.

To maintain the integrity of the WRR FPV input constraints, we conducted verification at both the unit level and in broader simulation (Cluster level) runs. It was imperative to transform all assumptions used at the WRR level into assertions in both unit-level FPV and simulation environments. This rigorous approach not only clarified any ambiguities in constraint coding but also revealed instantiation bugs within the unit-level WRR. We discovered that input violations to the arbiter were occurring, particularly when arbiters were configured in cascaded arrangement, either in parallel or serial, and the output from one arbiter instance served as input to another. Most of these constraint violations emerged under such cascaded configurations, highlighting the importance of meticulous verification across all levels of integration.

V. RESULTS

The verification efforts yielded exceptional results, uncovering approximately 50 critical bugs that were categorized into four distinct types, each representing a different aspect of the arbiter's functionality. The following outlines the categories and specific instances of the bugs identified through Formal Property Verification.

TABLE 2
CLASSIFICATION OF BUGS FOUND

Type	Example
Round Robin Algorithm Violations	An issue was detected where the internal grant priority pointer failed to update to the next requester during a multi-cacheline data sequence if an output stall was asserted simultaneously with the request being dropped. This resulted in the same requester erroneously receiving the subsequent grant
Starvation of Non-Diamond Lane Clients	The internal counter for Diamond Lane grants was found to be erroneously stuck, causing Non-Diamond Lane clients to be starved even in the absence of Diamond Lane requests.
Multi-Cacheline Feature Violations	During an output stall, the design was improperly masking only single cacheline write requests and not multi-cacheline writes, leading to the starvation of single cacheline write requests.
Starvation Due to Stalls at Output Interface	The design erroneously masked both write and read requests during multi-cacheline read-return data grants, contravening the meta-packing feature's intent to opportunistically grant reads, thereby causing read starvation.

These findings underscore the critical role of FPV in the verification process, as it enables the detection and rectification of complex issues that could compromise the arbiter's performance and reliability. The resolution of these bugs has significantly bolstered the robustness of the WRR arbiter, ensuring its proper function in high-performance computing environments.

VI. CONCLUSION

The intricate nature of the design, characterized by a multitude of corner cases and a shallow sequential depth, positions Formal Property Verification as the quintessential tool for achieving comprehensive state space coverage in the verification process. FPV has proven particularly adept at uncovering a variety of deadlock scenarios that might elude with other verification methodologies. Given that the Weighted Round Robin (WRR) arbiter is a fundamental component anticipated to serve across multiple iterations of the graphics IP, the FPV testbench developed for this project is poised to be an asset for future generations. It is designed to efficiently capture regression bugs with minimal manual intervention, thereby streamlining the verification workflow for subsequent feature enhancements and arbitration policy updates.

The bugs identified by FPV, have profound implications for the overall performance of the arbitration logic. These findings underscore the critical impact of FPV in detecting performance-related issues within arbiter designs, where each operational step is stringently defined by the specification. This work reaffirms the formidable capability of formal assertions not only in ensuring functional correctness but also in safeguarding the performance integrity of arbiter mechanisms, thereby reinforcing their indispensability in the design and verification of complex digital systems.

REFERENCES

[1] E. Seligman, T. Schubert and M.V.A. Kiran Kumar, Formal Verification: An Essential Toolkit for Modern VLSI Design, Morgan Kaufman, 2015.

[2] M, Achutha Kiran Kumar V; SS, Bindumadhava; Puri, Disha; Verma, Vichal, “Designer’s validation nightmares - Formal to the rescue: A GT Experience”, in DTTC 2019