Scaling Formal Verification of Network On Chip Using Path Decomposition

Bilal Ahmed[1], Umar Yaqoob[1], Bilal Zafar[1], Misbahud Din[2]
10xEngineers[1], LUMS[2], Lahore, Pakistan

Abstract— Simulation-based techniques, while widely used for verifying complex systems, struggle to handle the distributed, parallel, and reactive behavior of modern Network-on-Chip (NoC) architectures. This limits coverage and confidence in detecting critical behaviors such as deadlock, livelock, and starvation. Formal verification, though capable of exhaustive proofs, faces convergence barriers due to state-space explosion in large multi-router networks. To address this, we present a scalable formal verification methodology for NoCs that systematically proves the correctness of routing micro-architectures. The proposed hybrid flow combines compositional reasoning, packet abstraction and path decomposition to enable tractable formal verification of NoCs against a comprehensive property suite. Using path decomposition technique, we overcome the state-space explosion challenge and achieve complete proofs for all possible paths within practical runtime and memory limits. We verified the design against properties including data integrity, packet ordering, deadlock and livelock freedom, starvation freedom, and liveness of injection. The methodology employs a reduced mesh-based model that captures real-world traffic patterns and fault conditions while remaining feasible for formal tools.

I. Introduction

Formal verification has evolved from an academic concept into a mainstream industrial practice for validating complex hardware systems. As System-on Chips(SoCs) scale in size and heterogeneity, simulation-based verification alone is insufficient to ensure correctness across all operational scenarios. In contrast, formal methods produce rigorous proofs that provide high confidence in design correctness while enabling early detection of bugs and corner-case scenarios often missed by simulation. Consequently, leading semiconductor companies now integrate formal verification as a standard component of their verification flow.

A. Motivation

This work aims to improve the scalability of formal verification for NoC architectures, enabling completeness of proofs across practical NoCs. While simulation remains valuable for functional testing, it cannot guarantee the absence of rare concurrency induced bugs. Formal methods, though capable of exhaustive proofs, sometime struggle to converge when applied to complex or sizable NoC architectures. This work aims to bridge that gap through a scalable verification methodology. The proposed approach enables efficient verification of critical NoC properties such as data integrity, packet ordering, and forward progress [1] within practical computational bounds, moving toward feasible formal sign-off for complex interconnect architectures.

B. Challenges

Applying formal verification to NoCs introduces several practical challenges. The foremost is state-space explosion, as the number of reachable states increases exponentially with network size and buffering depth. Even modest meshes can exceed the computational limits of formal solvers, making end-to-end proofs intractable. As illustrated in Figure 1, both runtime and memory usage rise exponentially with proof depth, underscoring the scalability barrier for conventional formal approaches [2].

A second challenge stems from the distributed and reactive behavior of routers. Each router operates autonomously while coordinating with its neighbors, leading to cyclic dependencies that complicate verification of global properties such as deadlock or livelock freedom. These require temporal reasoning across multiple hops, further increasing the complexity.

Lastly, defining tractable yet meaningful properties is essential. Verifying NoC correctness involves proving safety and liveness properties, and proving them without over-constraining the model or compromising coverage requires precise properties and abstraction of the data path and control logic.

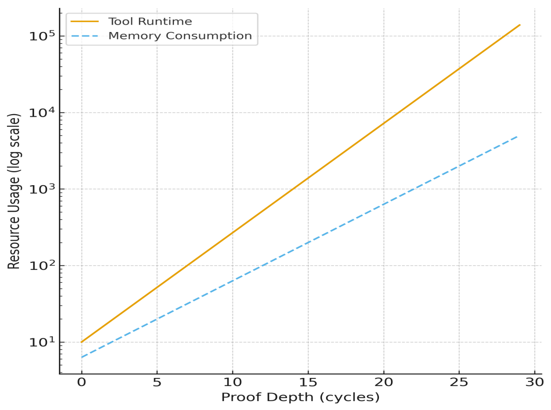


Figure 1. Memory usage and proof time grow exponentially with proof depth in direct formal verification.

C. Our Solution

Our verification methodology employs a combination of formal verification techniques, including compositional reasoning and end to end packet tracking to achieve complete and exhaustive verification of NoC. At the core of this methodology lies our path decomposition framework, which divides long communication paths into smaller, independently verifiable segments while preserving end-to-end correctness. This technique enables the packet tracker to scale seamlessly across the entire NoC, effectively proving all properties and ensuring comprehensive verification of NoC properties mentioned in Table 1.

Table 1. NoC Formal properties and their descriptions.

Property	**Detail**
In-order delivery	Packets from a given source to a destination arrive in the same sequence as injected.
Deadlock freedom	The network must never reach a circular dependency where packets cannot advance.
Livelock freedom	Packets must not circulate indefinitely without delivery.
Starvation freedom	Every packet must progress forward and eventually evacuate the NoC.
Liveness of injection	The system as a whole must make forward progress under continuous injection.
Evacuation	During backpressure, packets should still eventually drain from the network.
Local liveness	Each router should guarantee transmission of buffered packets when capacity is available.
Functional correctness	All packets must be routed to their correct destinations without corruption.

II. Background

A. NoC Architectures and Properties

As SoCs scale to include a growing number of heterogeneous processing elements, the NoC has emerged as the de facto interconnect fabric, replacing traditional shared buses and crossbars. A NoC provides a scalable, packet-switched communication backbone capable of sustaining concurrent transfers among multiple nodes with predictable latency and throughput. A 4x4 NoC in Mesh topology is shown in Figure 2. NoCs have become the standard on-chip communication fabric in modern many-core and heterogeneous SoCs, replacing traditional bus-based interconnects. They provide parallel communication among processing, memory, and peripheral components through a structured topology of routers and links. Typical NoC architectures employ deterministic or adaptive routing, virtual channels, and credit-based flow control to balance performance, deadlock avoidance, and implementation complexity. For the experimental validation in this work, we used BaseJump STL (BSG) NoC [3], an open-source and highly parameterizable mesh-based interconnect. This NoC's modular structure and configurability make it a practical platform for exploring scalable formal verification.

To ensure the reliability and correctness of a NoC, it is essential to verify not just local router behavior but also the emergent global properties that govern end-to-end communication. The key properties listed in Table 1, collectively capture the functional soundness and progress guarantees of the network. Proving them formally ensures complete and

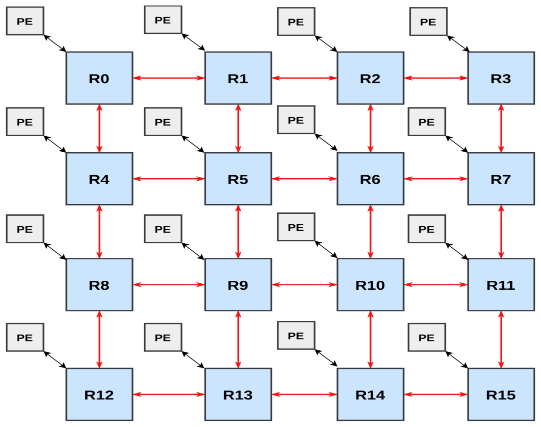


Figure 2. A 4x4 Mesh NoC with Processing elements connected to Routers.

exhaustive verification of the NoC [1], covering both correctness and liveness (guaranteed forward progress) across all communication paths [4]. Verifying these properties across all traffic patterns, routing decisions, and concurrent interactions is a formidable challenge, particularly when network size and configuration diversity are considered.

A1. Verification of NoCs (existing strategies).

Verification of NoCs has traditionally relied on simulation-based methods. These approaches employ directed and constrained-random testing to validate functional behavior and achieve coverage. However, due to the vast state-space and numerous corner-cases in NoC communication, simulation solely cannot achieve complete coverage. As a result, critical issues such as deadlocks, starvation, and routing anomalies may remain undetected until silicon bring-up.

Formal verification has become a complementing strategy to simulation in validating NoCs. Early work in this area focused on verifying individual router components such as allocators, arbiters, and virtual-channel controllers through property checking to ensure local correctness [5]. However, property verification for full NoC fabrics exposed severe scalability limits due to inter-router dependencies and concurrent traffic flows, leading to exponential state-space growth. To mitigate this, researchers introduced abstraction and compositional reasoning methods, including assume–guarantee reasoning [6], data and topology abstraction, symmetry reduction, and cut-based or bounded-environment decomposition [2]. Later approaches explored flow-level and credit-based abstractions to achieve liveness and deadlock freedom [7]. Despite these advances, achieving both scalability and completeness remains a major challenge, overly abstract models risk missing subtle corner cases, while complex models quickly become computationally infeasible.

While these methods have improved the tractability of NoC verification, they often struggle to capture end-to-end and liveness properties, such as starvation freedom and packet evacuation. Verifying these properties across multi-router networks requires scalable methodologies that preserve accuracy while overcoming the limitations of existing techniques.

III. Verification Strategy

Our formal verification methodology for NoCs is organized into two complementary stages: compositional reasoning and end-to-end verification. The first stage establishes local correctness of router components through property verification and/or modular abstraction. The second stage leverages these verified components to prove mesh-wide properties, ensuring packet-level correctness, evacuation, and liveness across the full NoC fabric. Together, these stages form a coherent and scalable verification framework that bridges router-level properties with packet evacuation guarantees throughout the mesh.

A. Compositional Reasoning

Compositional reasoning forms the technical foundation of our verification methodology. A router is composed of several submodules, such as FIFO, decoder, and arbiter. Figure 3 shows individual modules with the corresponding

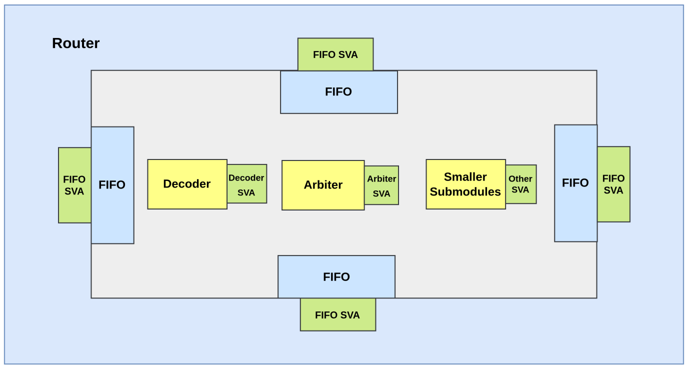


Figure 3. Compositional verification framework integrating individual router components with their dedicated testbenches.

testbenches encapsulated in the router. Each module is first verified in isolation, with properties based upon design intent. These properties capture essential control and data flow requirements, ensuring correctness of operations like buffering, routing decision, and arbitration. Once these proofs are established, they are assumed as guarantees at the next hierarchy level to verify router-level properties, including flow control, packet forwarding, and interface compliance [5]. This hierarchical proof composition significantly reduces the verification state-space, enabling formal convergence.

B. End-to-End Verification

While compositional reasoning ensures that individual routers behave correctly in isolation, it does not automatically guarantee that packets traverse the network correctly or that the overall system remains free from deadlock or starvation. To address this, we introduce an end-to-end verification framework that captures packet-level properties across the NoC fabric. This framework combines topological reduction, case splitting and symbolic tracking to prove packet integrity, assuring that packets injected at any source node are eventually delivered to their intended destination under all possible routing and arbitration scenarios.

B1. Parameter Reduction

Verifying a complex NoC mesh with hundreds of routers quickly become intractable for formal tools due to the combinatorial explosion of routing paths and state dependencies. We adopt a mesh reduction strategy, wherein the network is scaled down to a smaller but behaviorally similar mesh that preserves the essential communication patterns. A large 16×16 mesh can be reduced to a functionally equivalent 8×8 or 4×4 topology while preserving corners, edges, and inter router link capabilities. This reduction preserves architectural symmetry, enabling proofs on the smaller model to generalize to the full mesh. The approach enables exhaustive verification of global properties such as liveness, non-blocking progress, and deadlock freedom within tractable computational limits.

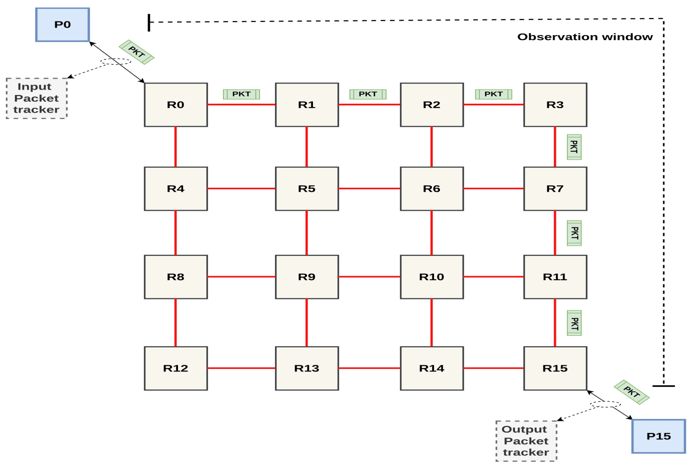


Figure 4. Packet tracking for path R0 to R15 visualized in a 4x4 mesh.

B2. Case Splitting

To further enhance tractability, we employ case splitting based on source-destination pairs. Each verification case isolates a source-destination path, while keeping the rest of the network out of the cone-of-influence(COI). This targeted decomposition reduces the size of the formal COI for each property, enabling faster convergence and parallel execution of multiple verification runs. By systematically covering all possible routing cases, this method achieves full coverage without redundant state exploration.

B3. Packet Coloring

Since the packet structure of the BSG NoC [3] does not include explicit source information, we introduced packet coloring, where each packet is assigned a unique color based on its source. This abstraction enables simultaneous tracking of multiple packets without interference among their behaviors. Packet coloring proved instrumental in verifying ordering guarantees, fairness, and non-interference across distinct traffic flows. Overall, it provides a lightweight yet powerful mechanism for modeling multi-router concurrence and resource isolation within NoC fabrics.

B4. Packet Tracker

The Smart Tracker technique, originally introduced in [9], provides an efficient means to symbolically track data as it propagates through communication pipelines. While highly effective for single-channel or point to point communication, its direct application to NoCs is challenging due to distributed nature, concurrent packet flows, and variable routing paths. To overcome these limitations, we extended the concept into a Packet Tracker abstraction, uniquely adapted for NoC verification. Figure 4 shows a visual representation of Packet tracker applied to the longest hop in a 4x4 mesh. We track a symbolic packet injected at the source router as it propagates to the destination router. This is made possible with packet coloring and case splitting to distinguish packets symbolically across multiple routing scenarios, allowing each communication path to be verified in isolation while preserving overall network correctness. By maintaining symbolic correspondence between packet identifiers, colors, and routing decisions, the Packet Tracker enables scalable proofs of packet delivery, ordering, and packet integrity across multi-hop NoC fabrics. The code snippet 5, shows implementation of packet tracker with coloring and case splitting applied to a 4x4 mesh verifying all possible 240 paths.

```verilog
generate
  for (i = 0; i < Mesh_size; i = i + 1) begin: src_loop
    for (j = 0; j < Mesh_size; j = j + 1) begin: dst_loop
      if (i != j) begin
        assign hsk_in[i][j] = Valid_in[i][P];
        assign hsk_o[i][j] = Valid_out[dest_rtr[i][j]][P] &
                             ready_out[dest_rtr[i][j]][P];
        assign dest_rtr[i][j] = watched_pkt[i][j][dest_bits:0];
        assign src_rtr[i]      = i[src_bits - 1:0];
        assign colored_pkt[i][j] = {src_rtr[i], watched_pkt[i][j]};
        assign ready_to_sample[i][j] = hsk_in[i][j] & readyin[i][P] &
                                       (data_i[i][P][width_lp-1:0] == watched_pkt[i][j]) &
                                       arbit_window;
        assign incr[i][j] = hsk_in[i][j] & !sampled_in[i][j] &
                             readyin[i][P] &
                             (data_i[i][P][dest_bits -1:0] ==
                              dest_rtr[i][j]);
        assign decr[i][j] = hsk_o[i][j] & !sampled_out[i][dest_rtr[i][j]] &
                             (data_o[dest_rtr[i][j]][P][src_bits + width_lp - 1:width_lp] ==
                              src_rtr[i]) &
                             (data_o[dest_rtr[i][j]][P][dest_bits - 1:0] ==
                              dest_rtr[i][j]);
        assign must_read[i][j] = (tracking_cnt[i][j] == 1) & sampled_in[i][j] & decr[i][j];
        data_integ: assert property (must_read[i][j] |-> (data_o[dest_rtr[i][j]][P] == colored_pkt[i][j
             ]));
      end
    end
  end
endgenerate
```

Figure 5. Example Verilog instrumentation for tracking forward progress.

Our approach avoids black-boxing router submodules, instead verifying end-to-end properties over the actual hardware. This preserves microarchitectural accuracy, enabling exhaustive yet realistic proofs of safety and liveness under true implementation semantics.

IV. Path Decomposition

Compositional reasoning and packet tracker abstraction works well for verifying routers and small meshes but struggle to scale with network size. Each router introduces additional buffers, arbiters, and control logic that interact combinatorially, causing verification complexity to grow exponentially with path length. To mitigate this, we propose path decomposition, a structured method that divides a longer path into smaller, verifiable segments while preserving global correctness of properties. Consider a packet traveling from source router R_0 to destination R_{15} through intermediate nodes. Instead of verifying the entire path $R_0 \to R_{15}$ as one monolithic model, we partition it into overlapping subpaths such as $R_0 \to R_2$, $R_2 \to R_7$, and $R_7 \to R_{15}$. Each subpath becomes an individual path for packet tracker to track, while keeping the semantics of the actual path intact. Typically subpath is divided into two hops which the model checker can exhaustively explore all reachable states and prove local properties like routing correctness, buffer consistency, and forward progress. A visual example of a 4 hop path segmented using path decomposition is shown in figure 6.

Formally, let each subpath be defined as:

$$P_i = (R_{s_i} \to R_{d_i}),$$

Where R_{s_i} and R_{d_i} refer to source and destination of a subpath P_i. So the global path P as an ordered composition:

$$P = P_1 \wedge P_2 \wedge P_3 \wedge \cdots \wedge P_n,$$

The terminal state of P_i serves as the initial state of P_{i+1}. By proving that each segment preserves the packet's routing intent and data integrity, we ensure that their composition maintains the same invariants for the entire path.

$$G(P) = G(P_1) \wedge G(P_2) \wedge G(P_3) \wedge \ldots G(P_n)$$

Thus, the global correctness property $G(P)$ remains valid if and only if all local properties $G(P_i)$ hold under consistent boundary assumptions. This approach transforms the exponential growth of verification complexity into a linear relationship with the number of segments. If the average verification time for a two-hop subpath is T_2, then the total verification time for a path of k hops can be approximated as:

$$T_k \approx \frac{k}{2} \times T_2.$$

This near-linear scaling enables exhaustive proofs for significantly larger topologies up to 16×16 meshes without loss of logical completeness or behavioral accuracy.

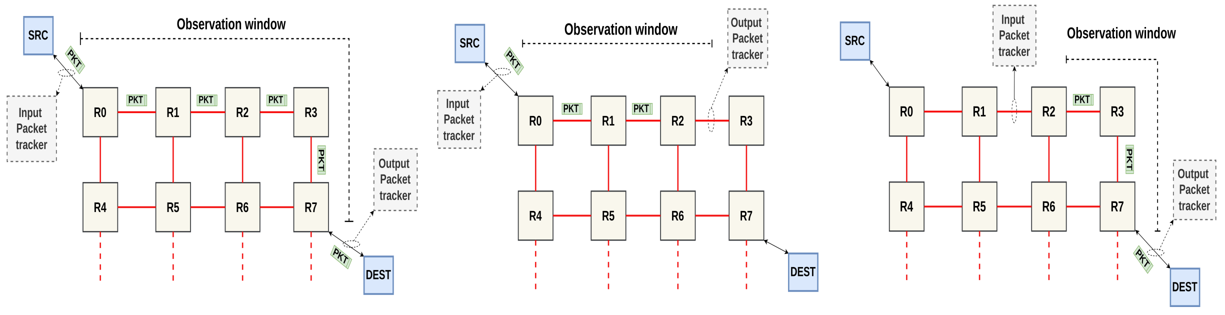


Figure 6. Visual representation of path decomposition of a 4 hop path into two, 2 hop segments.

A. Path Segmentation Modeling

We model the routing algorithm of the NoC to accurately predict the complete packet route at the source. This predictive model, implemented in Python, automatically segments end-to-end communication paths based on source–destination pairs. Each segment is constructed such that the destination of one subpath overlaps with the source of the next, ensuring continuity and proof completeness across the entire route. The generated path segmentation remains consistent with the mathematical model presented earlier. Under the path decomposition framework shown in Figure 6, each segment is independently verified to establish full end-to-end correctness across the full path.

B. Applications

In addition to enabling scalable end to end packet integrity, this technique was also instrumental in proving liveness properties across all possible communication paths. By combining this method with the Packet Tracker abstraction, we ensured that every packet injected into the network was delivered to its intended destination and also guaranteed forward progress under all permissible routing scenarios. This integration allowed us to formally establish both safety and liveness proofs within the same verification framework, demonstrating that the proposed approach effectively scales to full-network verification without compromising completeness.

Importantly, most commercial NoCs employ source-based routing due to its simplicity, predictability, and verification efficiency. As a result, the proposed technique is broadly applicable to practical designs, providing a scalable and reusable verification framework for modern NoCs with statically determined routing. However, the technique does not directly apply to dynamically routed networks.

The path decomposition technique is particularly suited to NoCs with deterministic routing such as source or dimension-ordered routing (DOR) [10] where the full packet path is known before transmission. This predictability enables formal modeling of the routing algorithm and systematic path segmentation without losing end-to-end correctness. We applied this approach to both the BSG NoC [3] and FLooNoC [8], exploiting their deterministic routing behavior to verify smaller subpaths that together guarantee global integrity.

V. Results

This section presents the experimental validation of the proposed formal verification methodology on the BSG NoC [3] architecture. We analyze convergence, runtime, and memory utilization across multiple mesh sizes to assess scalability. While conventional formal techniques typically fail to converge beyond short communication distances, our approach driven by path decomposition successfully extends end-to-end verification to larger NoC configurations, maintaining proof completeness and design accuracy throughout.

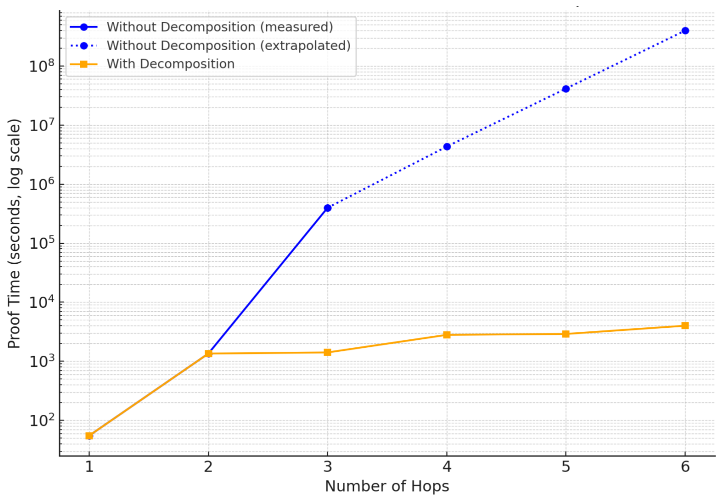


Figure 7. Formal proof runtime against number of hops for a 4x4 mesh.

Employing path decomposition led to a dramatic improvement in solver convergence, achieving up to a $10^5\times$ speedup for six-hop communication paths, as illustrated in Figure 7. The speedup was estimated by extrapolating baseline runtime from three hop proofs, which exhibited super-exponential growth. The improvement was consistent across both X–Y and Y–X routing configurations, demonstrating the robustness of the path decomposition framework. By segmenting complex paths into verifiable subpaths, we successfully proved all 240 possible source-destination pairs in a 4×4 mesh within 50 hours, while liveness property proofs covering deadlock freedom, starvation freedom, and forward progress converged within 72 hours. The experiments were conducted using Cadence Jasper (2025.06) on an Intel Core i5 (8th Gen) processor with 16 GB RAM, showcasing that exhaustive NoC verification can be achieved even with modest computational resources. Beyond successful proof convergence, the framework also achieved formal coverage closure across all evaluated NoC configurations.

We achieved complete formal verification for all 4×4 mesh configurations, including simple, half-ruche, and full-ruche topologies, attaining over 99% formal coverage within practical runtimes. For larger 8×8 meshes, our framework successfully verified the majority of safety and liveness properties, reaching 90% formal coverage(due to incomplete exclusions), despite a significant increase in design complexity and proof depth as shown in Table 2. This level of completeness enabled us to uncover and report the design bugs discussed in the following sections.

Table 2. Formal verification results and design complexity across NoC topologies.

Design	Topology	Routers	Channels	FIFOs	Flops	Gates	RTL Lines	Assertions	Assertions Runtime	Formal Coverage
Simple Mesh 4x4	Mesh	16	80	80	8585	292k	3558	2128	72h	99%
Half Ruche 4x4	Torus	16	112	112	9283	322k	4385	3106	65h	99%
Full Ruche 4x4	Full Torus	16	144	144	10944	360k	5299	4112	57h	99%
Simple Mesh 8x8	Mesh	64	320	320	302k	1701k	3652	14336	202h	90%

A. Bug Examples

During formal verification of the BSG NoC [3] and FlooNoC [8], some subtle design bugs were uncovered in some configurations of the design, each exposing a unique aspect of microarchitectural behavior that traditional simulation could easily overlook. These issues highlight the strength of exhaustive formal analysis in ensuring correctness across all reachable states.

A1. Packet Dropping Scenario

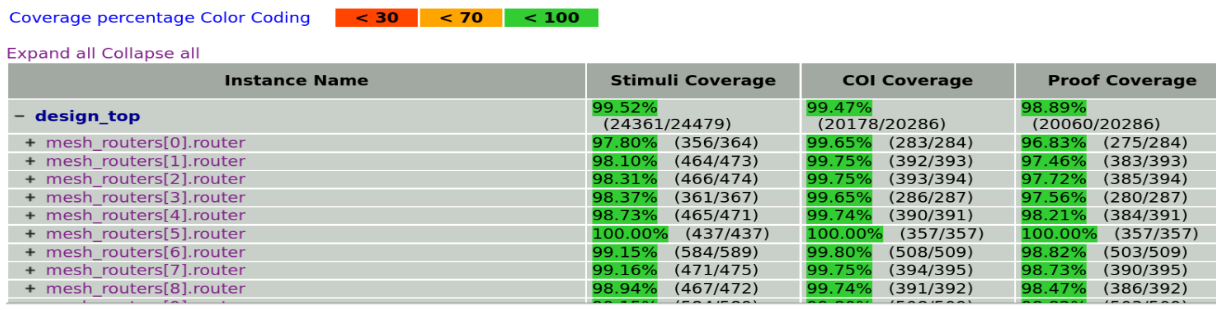
One class of bugs involved packet loss due to improper synchronization between input FIFO readiness and output valid signals. Under certain timing alignments, a packet would be dequeued but never forwarded to the downstream router, effectively disappearing from the network. The formal tool generated a precise counterexample showing the specific sequence of valid/ready transitions leading to this condition, enabling targeted fixes in the flow-control logic.

A2. Arbitration Errors

Another issue was found in the routing computation, where certain destination coordinates triggered incorrect output port selection. This occurred in rare corner cases where route computation and buffer arbitration overlapped, leading to packets being misrouted. The counter examples exposed this during the verification process.

B. Formal Coverage

Formal coverage is a metric very similar to simulation-based coverage with the same goals, to measure the completeness of the formal verification work. Formal coverage for a 4x4 mesh of BSG NoC was calculated with the proposed verification strategy. After excluding dead code and logic related to inactive boundary channels, we achieved high formal coverage across all active modules as shown in figure 8. This demonstrates that the proposed verification methodology ensures exhaustive property checking but is also capable of achieving full coverage closure for NoC designs.


Coverage percentage Color Coding 🟥 < 30 🟧 < 70 🟩 < 100

Expand all Collapse all

Instance Name	Stimuli Coverage	COI Coverage	Proof Coverage
− **design_top**	99.52% (24361/24479)	99.47% (20178/20286)	98.89% (20060/20286)
+ mesh_routers[0].router	97.80% (356/364)	99.65% (283/284)	96.83% (275/284)
+ mesh_routers[1].router	98.10% (464/473)	99.75% (392/393)	97.46% (383/393)
+ mesh_routers[2].router	98.31% (466/474)	99.75% (393/394)	97.72% (385/394)
+ mesh_routers[3].router	98.37% (361/367)	99.65% (286/287)	97.56% (280/287)
+ mesh_routers[4].router	98.73% (465/471)	99.74% (390/391)	98.21% (384/391)
+ mesh_routers[5].router	100.00% (437/437)	100.00% (357/357)	100.00% (357/357)
+ mesh_routers[6].router	99.15% (584/589)	99.80% (508/509)	98.82% (503/509)
+ mesh_routers[7].router	99.16% (471/475)	99.75% (394/395)	98.73% (390/395)
+ mesh_routers[8].router	98.94% (467/472)	99.74% (391/392)	98.47% (386/392)

Figure 8. Coverage details for a 4x4 Mesh BSG NoC.

VI. Future Works

We aim to extend this verification methodology to cover adaptive routing schemes and unique NoC topologies, exploring its applicability to diverse industrial implementations. Additionally, we plan to further scale the approach to larger network fabrics, enabling complete formal verification of high-performance, many-core NoC architectures.

Acknowledgments

The authors gratefully acknowledge the School of Science and Engineering(SSE), Lahore University of Management Sciences (LUMS) for providing the infrastructure and research environment essential for this work, which was instrumental in conducting the experiments presented in this study.

References

[1] F. Verbeek, "Formal Verification of On-Chip Communication Fabrics," Ph.D. thesis, 2013.

[2] N. D. Kim, J. Park, H. Singh, and V. Singhal, "Sign-off with Bounded Formal Verification Proofs," Samsung Electronics, Giheung, South Korea, and Oski Technology, Gurgaon, India and Mountain View, CA, USA.

[3] Basejump STL, `https://github.com/bespoke-silicon-group/basejump_stl`

[4] A. Garg, "Forward Progress Checks in Formal Verification: Liveness vs Safety," *DVCon*, 2024.

[5] P. Roy, P. Yeung, J. Hong, A. Desai, A. Raj, C. Agarwal, and D. Patel, "Hierarchical Formal Verification and Progress Checking of Network-On-Chip Design," *DVCon US*, Nvidia Corp., Santa Clara, USA and Gurugram, India, 2025.

[6] D. Giannakopoulou, C. S. Păsăreanu, and J. M. Cobleigh, "Assume-Guarantee Verification of Source Code with Design-Level Assumptions," NASA Ames Research Center, Moffett Field, CA, and University of Massachusetts, Amherst, USA.

[7] A. Zerdani and F. Boutekkouk, "An Overview of Formal Verification of Network-on-Chip (NoC) Methods," in *ICISAT 2023*, M. R. Laouar, V. E. Balas, V. Piuri, D. Rad, Z. Touati Hamad, and A. Cheddad, Eds., .

[8] T. Fischer, M. Rogenmoser, M. Cavalcante, F. K. Gürkaynak, and L. Benini, "FlooNoC: A Multi-Tb/s Wide NoC for Heterogeneous AXI4 Traffic," *IEEE Design & Test*, Dec. 2023. DOI: 10.1109/MDAT.2023.3306720.

[9] A. Darbari and I. Singleton, "Industrial Strength Formal Using Abstractions," Imagination Technologies.

[10] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*, Morgan Kaufmann Publishers, 2004, ISBN 0-12-200751-4.