Visual Exploratory Analysis for Designing Large-Scale Network-on-Chip Architectures: A Domain Expert-Led Design Study

Shaoyu Wang[1*], Hang Yan[1*], Katherine E. Isaacs[2], Yifan Sun[3]
[1] Huazhong University of Science and Technology [2] The University of Utah [3] William & Mary

Abstract—Visualization design studies bring together visualization researchers and domain experts to address yet unsolved data analysis challenges stemming from the needs of the domain experts. Typically, the visualization researchers lead the design study process and implementation of any visualization solutions. This setup leverages the visualization researchers' knowledge of methodology, design, and programming, but the availability to synchronize with the domain experts can hamper the design process. We consider an alternative setup where the domain experts take the lead in the design study, supported by the visualization experts. In this study, the domain experts are computer architecture experts who simulate and analyze novel computer chip designs. These chips rely on a Network-on-Chip (NOC) to connect components. The experts want to understand how the chip designs perform and what in the design led to their performance. To aid this analysis, we develop Vis4Mesh, a visualization system that provides spatial, temporal, and architectural context to simulated NOC behavior. Integration with an existing computer architecture visualization tool enables architects to perform deep-dives into specific architecture component behavior. We validate Vis4Mesh through a case study and a user study with computer architecture researchers. We reflect on our design and process, discussing advantages, disadvantages, and guidance for engaging in a domain expert-led design studies.

Index Terms—Data Visualization, Design Study, Network-on-Chip, Performance Analysis

1 Introduction

Complex analysis can benefit from interactive visualization support, but often existing visualization tools and design knowledge are not sufficient for the specific needs of experts in a particular domain. Visualization design studies [49] are an avenue by which visualization researchers can better understand such scenarios and develop new knowledge about visualization design, techniques, and methodology while also helping their domain collaborators.

In a typical design study, a visualization team takes the lead in designing the visualization system and gathers preliminary data and evaluation of the system through interactions with the domain experts. The frequency of these interactions varies and can be a point of tension in the design process [40]. Furthermore, when the visualization team implements the system apart from the domain experts, we as a community gain little insight into the barriers that prevent domain experts from successfully developing their own visualizations. While for some domain expert groups these barriers are clear, such as when solutions require computer programming skills, this is not the case in domains where such skills are prevalent. To explore these issues, we conduct a design study in the domain of computer architecture, structured such that the domain experts lead the design and implementation of the visualization.

Our study addresses challenges in analyzing simulation data during computer chip design. To evaluate, understand, and modify their design, they simulate their design under workloads (i.e., as generated by computer programs) and analyze the simulated data to explore how the chip may behave in reality. As chips grow in both computing cores and memory units, the network connecting those components, the *Network-on-Chip (NOC)* becomes a central actor in the chip's performance. Thus, computer architects need to understand not only what occurs in the computing cores and memory, but also through the NOC.

Through regular meetings between the domain team and a visualization expert, we design an interactive multi-view visualization system, Vis4Mesh, to enable computer architects to explore simulated NOC designs. Vis4Mesh supports both temporal and spatial tasks in exploring simulated NOC data and supports bidirectional synchronization with an existing state-of-the-art visualization tool for deep dives into data of additional architectural components. Our preliminary evaluations through user sessions and a case study show Vis4Mesh is capable of helping computer architects make sense of hardware behavior and identify performance bottlenecks.

Throughout our study, we reflect on the challenges and benefits of our domain expert-led process. We note tensions for domain experts in applying existing guidance, such as performing task analysis, differing perspectives of research contributions, and visualization authoring hurdles. We also identify advantages, such as familiarity with the data and potential for a tight implementation loop and integration with existing workflows. Finally, we discuss what aspects of the people and process of this collaboration enabled a successful design and design study. Our analysis provides insights on how the community can address the "design in the wild" problem [64].

In summary, this paper contributes:

- A domain analysis for network-on-chip visualization through informal observation and a literature review of the flagship NOC conference,

[*] *The two authors contributed equally.*
Manuscript received xx, 2023; revised xx, 2023.

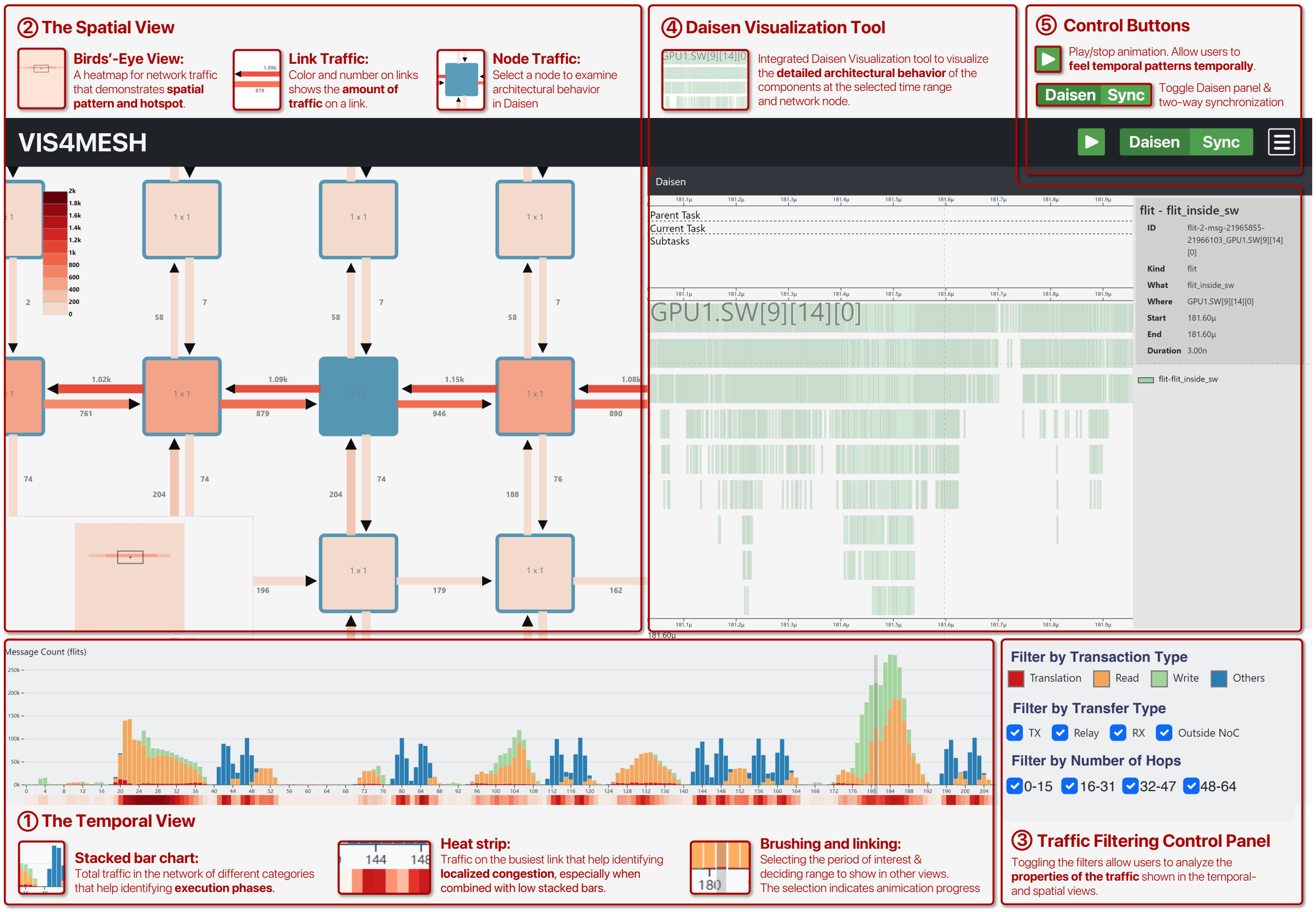

② The Spatial View
Birds'-Eye View: A heatmap for network traffic that demonstrates spatial pattern and hotspot.
Link Traffic: Color and number on links shows the amount of traffic on a link.
Node Traffic: Select a node to examine architectural behavior in Daisen
④ Daisen Visualization Tool
Integrated Daisen Visualization tool to visualize the detailed architectural behavior of the components at the selected time range and network node.
⑤ Control Buttons
Play/stop animation. Allow users to feel temporal patterns temporally.
Daisen Sync
Toggle Daisen panel & two-way synchronization
VIS4MESH
Daisen
Parent Task
Current Task
Subtasks
GPU1.SW[9][14][0]
flit - flit_inside_sw
Message Count (flits)
Filter by Transaction Type
Translation
Read
Write
Others
Filter by Transfer Type
TX
Relay
RX
Outside NoC
Filter by Number of Hops
0-15
16-31
32-47
48-64
③ Traffic Filtering Control Panel
Toggling the filters allow users to analyze the properties of the traffic shown in the temporal- and spatial views.
① The Temporal View
Stacked bar chart: Total traffic in the network of different categories that help identifying execution phases.
Heat strip: Traffic on the busiest link that help identifying localized congestion, especially when combined with low stacked bars.
Brushing and linking: Selecting the period of interest & deciding range to show in other views. The selection indicates animication progress

Fig. 1. An overview of Vis4Mesh. Vis4Mesh integrates highly coupled Temporal View, Spatial View, and Daisen Visualization tool, representing the temporal, spatial, and logical organization of the behavior of Network-on-Chip systems and the hardware components connected by the network.

- The design and implementation of Vis4Mesh, a validated visualization design which provides interactive temporal and spatial overviews for mesh NOC architectures, and
- Reflections on our domain expert-led design study process with implications for future such design studies.

2 Background

A *Network-on-Chip (NOC)* system integrates multiple components of a computing system by allowing them to exchange messages. The components are called *clients* and include microprocessors, accelerators (e.g., audio/video decoders, neural processing units), caches, memory controllers, input/output devices (e.g., Ethernet adapters), and persistent storage (e.g., solid-state drives).

The network comprises three fundamental components: adapters, switches, and links. *Adapters* are interfaces between clients and the network. *Switches* receive and forward data to adapters or other switches based on the destination of the data and the switch's routing rules for that destination. *Links* are wires connecting a switch to an adapter or switch, responsible for passing messages from one end to the other.

Data is sent over the network in *messages* (also called *packets*), which can be arbitrarily long. Typically, messages need to be subdivided into *flits*, the amount of data that can be transferred each clock cycle on each link. The flits then traverse multiple switches to reach their intended destination.

As switches and links have limited capacity, heavy network traffic (e.g., many flits passing through a switch) can cause delays as flits have to wait in queues rather than being sent to their destination immediately. We call this *network congestion*. Identifying and understanding reasons for congestion is an important task in designing NoC systems. Additionally, delays may be caused by limited buffer space for the flits at the source or destination adapter.

A *mesh* topology connects switches in a grid. It is one of the most popular NOC topologies (see our survey in subsection 5.2). The flat nature of silicons lead to 2D layouts of the mesh. Therefore, we focus on 2D mesh topologies in this work.

3 Related Work

We discuss related work in visualizing computer networks and domain expert visualization collaborations.

3.1 Visualizing Network on Chip Systems

Visualization of data-transfer networks has been an extensively studied domain in both the computing systems [2], [6], [65] and data visualization communities [15], [21]. Here, we focus on network-on-chip systems.

Visualization tools for NOC systems have primarily come from the NOC research community. Most of the existing tools assume mesh networks and use color encoding to demonstrate the amount of traffic on each component. NOCScope [42] shows the traffic going through each switch and each switch port on a mesh network. NocVision [16] lets users show the average, minimum, or maximum traffic on each link in each period. Additionally, VisualNoC [61] provides a NOC visualization tool specific for mesh-based accelerators (e.g., Google TPU [23]). VisualNoC highlights the capability of visualizing the traffic going through each switch and how the mapping algorithm (i.e., algorithms decide which core executes each piece of the program) impacts the system performance. Overall, existing tools provide a good starting point for the design of Vis4Mesh and define the "must-have" features of NOC visualizations. However, these tools have limited capability in visualizing application behavior along with its architectural causes, constraining users' capability of digging deep into the root cause and confirming their findings from another aspect.

Mesh and torus networks are also prevalent in high-performance computing (HPC) systems (i.e., supercomputers). Several works [9], [20], [21], [29], [55] visualize the traffic and congestion for 2D or 3D mesh and torus HPC interconnection networks. Unlike NOC systems, HPC interconnection networks also include higher dimension tori, requiring slicing [39] or projections [13], [59]. Fujiwara et al. [15] focuses on improving scalability with an analytics solution to show routing possibilities. These papers focus on an already-designed network between computing nodes and treat traffic as homogeneous. In contrast, chip design is an integrated process involving the network, computing elements, memory systems, and scheduling algorithms. Thus, the design of different systems is needed to support NOC system design-specific tasks, such as understanding traffic types while removing unneeded features like high-dimensional network support.

3.2 Domain Collaborations in Design Studies

There are a variety of techniques and guidance for improving domain understanding in visualization design studies, such as collaborative workshops [25], [26], [30], [41] and immersion in the domain workflow [14], [19]. In all of these strategies, ownership of the visualization design and implementation lies with the visualization experts. In contrast, in this project, the domain experts owned the visualization design and implementation while the visualization expert served in a consulting and guidance role.

Simon et al. [51] discuss a *Liaison* role for design studies, a person who speaks both the domain and visualization languages and thus acts as a bridge between domain experts and visualization designers. The domain and visualization experts in this project had experience with both languages, following patterns of knowledge gain similar to those described in becoming a liaison. This preparedness for communication further enabled the domain expert-led design process.

4 Design Process and Team

We followed an iterative design process informed by the design study methodology (DSM) [49] but with a key choice: the visualization design and implementation were led by architecture domain experts rather than visualization experts.

Our design team was composed of a computer architecture domain expert ("ChipExpert", 8-10 years computing systems research experience), a visualization expert ("VisExpert", 10+ years visualization research experience), and two computer architecture trainees (undergraduate students with 2-3 years computer architecture research experience) mentored by the ChipExpert. We refer to the three people from the architecture domain as the "ChipTeam."

All members of the ChipTeam were involved in interpreting data generated from the multi-GPU simulator, MGPUSim [56], which means they could also be viewed as *front-line analysts* via DSM. We note that in academic computer architecture research, there is a need to both build and use tools, such as MPGUSim; so while the domain experts were also tool-builders, a significant portion of their work is as front-line analysts.

While the ChipExpert considers themself a computer architecture researcher, they had experience during their graduate studies designing a visualization system (as first author) and aiding in some human-centered computing projects of their peers (as co-author).

The project was initiated by the ChipExpert, who began guiding the trainees. Early in the project, when only the initial detail view had been prototyped, the ChipExpert reached out to the VisExpert for guidance because of the VisExpert's previous experience with computing network visualization. The collaboration continued with the full team meeting bi-weekly and the ChipTeam meeting weekly.

The ChipExpert led the project, managing the trainees' time and having the most influence on the design and prioritization of features and evaluation in consultation with the full team. The trainees implemented data processing and visualization tools.

The VisExpert provided guidance and feedback regarding the visual design, the task analysis, and the evaluation. Throughout the project, the VisExpert created memos regarding observations of interest to the greater visualization community, which are discussed in section 9.

5 Task and Data Abstraction

We present the task abstraction, data abstraction, and design requirements, after discussing our processes for eliciting them.

5.1 Task and Data Abstraction Process

The initial tasks and data abstractions came from ChipExpert's personal research experience, including observations of industry and academic practices and the demand of ChipExpert's research team. When the VisExpert joined, they argued for a broader source of tasks. Through dialogue with ChipTeam, ChipExpert also helped refine and codify the existing tasks.

To validate the ChipExpert's tasks and possibly identify additional tasks, the ChipExpert and VisExpert discussed other task elicitation techniques such as surveys, interviews, and literature searches. The previous survey conducted by the ChipExpert [58] had focused on architecture simulation analysis in general. As Vis4Mesh targets a more specific problem, we ultimately conducted a NOC-focused literature review.

5.2 NOC Literature Review

To better understand the tasks of computer architects while designing NOC systems, the ChipExpert performed a literature review (see Figure 2) of 32 papers published in the most recent three years' IEEE/ACM International Symposium on Networks-on-Chip (NOCS), which is a flagship conference in NOC design. Particular focus was given to how authors graphically depict NOC system issues and mechanisms and the metrics they use to demonstrate the effectiveness of designs. While other computer

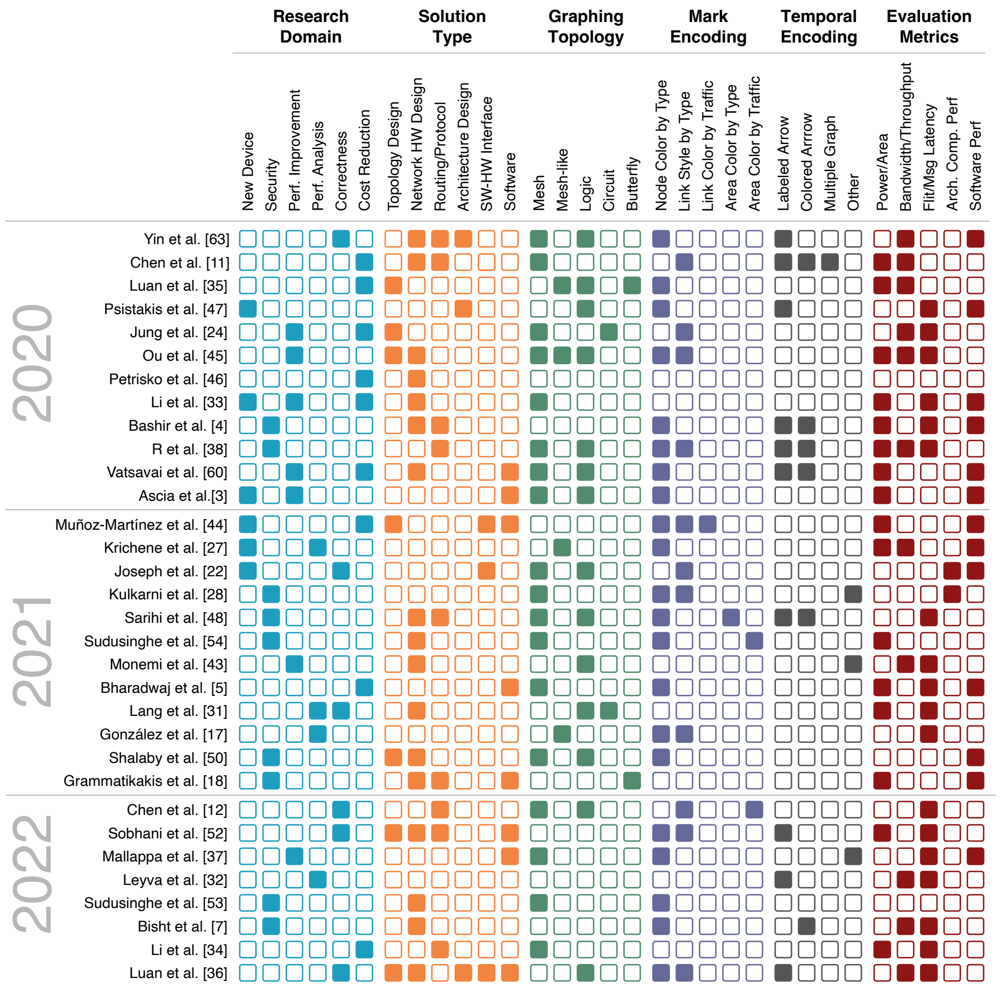


Fig. 2. Themes and individual codes from our literature review of all the papers [63] [11] [35] [47] [24] [45] [46] [33] [4] [38] [60] [3] [44] [27] [22] [28] [48] [54] [43] [5] [31] [17] [50] [18] [12] [52] [37] [32] [53] [7] [34] [36] published in the most recent three years' IEEE/ACM International Symposium on Networks-on-Chip (NOCS). Papers exhibited a variety of analyses between the NOC and architecture and software workload concerns, including temporal trends and metrics we use to validate and extend our task analysis and visualization design.

architecture conferences feature NOC papers, we aim to systematically summarize the visualizations used in communications among NOC-focused experts. After analyzing three years of papers, the ChipExpert concluded they were not seeing additional kinds of visualizations and metrics, and thus, the three-year sample represented the space. We discuss how we derive tasks and design guidance from the literature review below.

Mesh and mesh-like topologies dominated (64% of papers), so we decided to focus on mesh-like NOCs. While most papers demonstrated results with fewer than 100 components, there was a trend towards larger NOCs, suggesting the need for scale.

Many papers (43%) have figures showing temporal behavior, often with arrows. At least one used multiple figures to explain temporal behavior. This observation affirms the temporal tasks we identify.

We observed a strong coupling between NOC system designs, the architecture of the components (e.g., caches, cores), and the software workloads (see themes of Research Domain, Solution Type, and Evaluation Metrics in Figure 2). We thus affirm and expand tasks relating architecture and workload to NOC design.

NOCs are typically depicted as node-link diagrams, so we adopted this familiar idiom. The papers typically used color to indicate the component type. Sometimes, areas of congestion were also shaded. Traffic was shown through traditional line and bar charts. We suspect these choices are due to the expectation of statistical charts in performance results and the unavailability of interactivity. Thus, we choose color for traffic, an encoding successfully used by other non-architecture computer network visualizations in the related work.

5.3 Task Analysis

From the project inception, ChipTeam voiced their high-level goal of understanding the root causes of network congestion stemming from NOC and overall GPU design. The need for spatial and temporal views was more intuitively understood as they began their design, as congestion exists in both space and time.

The VisExpert helped refine the tasks, their context regarding goals, and workflow throughout the design process. We present the refined tasks and their lower-level sub-tasks.

Goal: To identify performance improvement opportunities. Computer architects aim to improve performance by identifying the causes of underutilized resources and then altering the design. Both network congestion and network under-utilization can be symptoms that stem from either network or non-network components. Therefore, computer architects analyze congested and non-congested periods to investigate fundamental issues rooted in the non-network subsystem.

T1: Understand the simulation workload. Computer architects need a high-level understanding (i.e., overview) of behavior under the simulated workload, including phases with different characteristics, as each may have different optimization concerns. These include types of messages (e.g., control versus data) and their flow patterns in the network (e.g., one component broadcasting to many others, long-distance paths).

T1.1 Identify workload phases.

T1.2 Characterize workload phases.

T1.1.1 Identify types of messages sent

T1.1.2 Identify high-level traffic flow patterns (e.g., north half is busy, but the hot area moves south over time)

T2: Identify congestion. Congestion must be identified to understand its cause. Both low and high message volumes are of interest as low traffic can indicate traffic blockage inside switches, possibly due to architecture choices.

T2.1 Locate congestion in time.

T2.1.1 Identify time periods with extreme (low or high) overall traffic load.

T2.1.2 Identify time periods with saturated tiles or links.

T2.2 Locate congested regions of the network.

T3: Assess network design effects on traffic. Poor performance may be due to sub-optimal network design, requiring the computer architect to understand how the network design affects traffic. Note that just as in the overview sub-tasks of T1, message flow patterns in the network are a key task.

T3.1 Identify types of messages involved in congestion

T3.2 Identify low-level message flow patterns (e.g., data scatters from a certain node)

T4: Assess architecture effects on traffic. All the messages transferred on the network originate from non-network components. Therefore, the root cause may be an unexpected behavior of a non-network component.

T4.1 Reason why the messages are sent.

T4.2 Identify software/hardware elements that generate the messages.

5.4 Data Abstraction

We cast the NOC as a directed network with a numeric attribute for traffic associated with each node and link. The traffic attribute is derived from event sequences of flits at each component. Thus, the value can change by filtering flits by type or time range. The attributes associated with the flits are, in part, designed by our team and summarized in Table I.

Additionally, each node (architecture component) in the network (NOC) also has an associated log of processes—an event sequence with durational events. These are generated and presented by a separate tool that we integrate.

Data Design. One advantage of including domain experts in the research team is the full control of the ecosystem that involves both the data generation, collection, and visualization process. Therefore, we define our own data trace format, considering a balance between the expressiveness of the data, simulation performance, and visualization performance. For example, instead of recording the event sequence of each flit, which would produce too much data and is not necessary for our common tasks, we instead record the number of flits for each type that passes through a component at each microsecond. We discuss similar choices in subsection 6.3.

TABLE 1
Trace format generated by the instrumented simulator.

Field	Format	Description	Example
Time	Time	Time step of the flits	2.3μ s
Src	Coordinates	Source tile location	(2, 4)
Dst	Coordinates	Destination tile location	(3, 8)
Msg	Text	The message type	“Read”
Trans	Text	The transfer type	“Relay”
Hops	Range	The total hops	“32 - 47”
Count	Number	Number of flits of this type	12

The traces are stored as JSON files. We chose file-based tracing over databases to eliminate indexing overhead and use directory and file naming along with data redundancy to preserve query performance.

The ChipTeam implemented the trace collection as a Go library and connect it to MGPUSim, a state-of-the-art GPU system simulator, using the underlying tracing mechanism of the Akita computer architecture simulator framework [56]. Minor modifications were made to MGPUSim to invoke the tracer’s API. The data format is not coupled with Vis4Mesh nor MGPUSim, allowing people to either implement their own data collector on other simulators or their own analysis tool for the data.

5.5 Design Requirements

We summarize a list of design requirements (**DR**) for Vis4Mesh. These design requirements are summarized from the research experience of the ChipTeam, existing network visualization tools, and the NOCS literature review.

DR1: The visualization should enable users to observe spatial utilization trends. Users should be able to answer questions such as “which link is likely to be a global bottleneck?”

DR2: The visualization should depict temporal trends of network utilization. Users should be able to answer questions such as “during which period is the NOC underutilized?” DR1 and DR2 should be simultaneously supported so users can quickly identify points of interest in time and space.

DR3: NOC visualizations must support cross-layer reasoning. Users should be able to answer questions such as “What are the computing cores doing, and is it causing congestion?”

DR4: The visualization should support the scale of next-generation NOC systems that involve thousands of nodes.

6 VIS4MESH

We design Vis4Mesh based on our collected tasks and design requirements. We followed an iterative process, with weekly meetings among the ChipTeam and biweekly meetings with the VisExpert to refine the design.

The resulting design is a coordinated multi-view system (see Figure 1). The spatial view represents the full mesh network, allowing users to view traffic along the links and switches within a time slice. It also supports temporal animation. The temporal view summarizes traffic behavior over time in multiple ways and allows time slice selection. Multiple filters allow the examination of specific kinds of messages and flits. Finally, features and insights

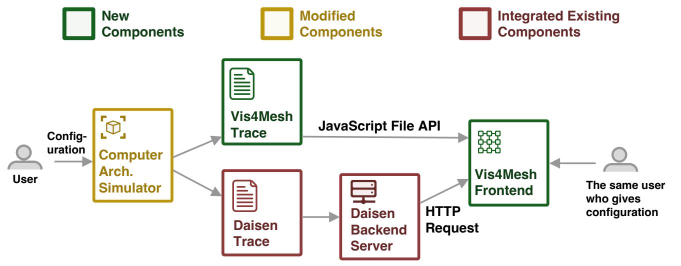


Fig. 3. The overview of the software components involved in Vis4Mesh.

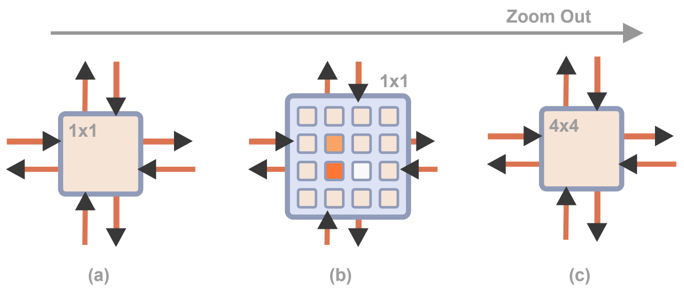


Fig. 4. Tile-aggregation-based semantic zooming design in the Detail View. When zooming out, every 16 single tiles (a) are aggregated into a single 4x4 tile (c). If further zoomed out, 4x4 tiles can also be aggregated into 16x16 tiles (no limitation). Also, to avoid confusion, we create a transition view (b) showing both the individual and aggregated tiles so that (a) and (c) smoothly blend into each other.

discovered in these views can be further investigated through integrated visualizations of specific architecture components and scheduling behavior from Daisen, a lower-level architecture visualization tool. Vis4Mesh is a web-based tool built with D3js [8]. It supports both in-situ and post-processing visualization. Figure 3 summarizes Vis4Mesh and its underlying components.

6.1 Visualizing Spatial Traffic Distribution

Revealing the spatial features of traffic (e.g., on which link, on which switch, in which area) is essential in understanding network performance (**T1.1.2**, **T2.2**, **T3**). Vis4Mesh provides two views that reveal the spatial distribution of the traffic (Figure 1 ②, the Spatial View), including the detailed view and the bird view, to fulfill **DR1**.

The Detail View. The spatial view of Vis4Mesh is a representation of the switches and links in the network, allowing users to see traffic information down to individual network components. We represent the full mesh network through a regular grid of square tiles ▢ connected by arrows ➞ representing links. Each tile has eight arrows for its incoming and outgoing communication with its four neighbors (east, south, west, and north).

Traffic across each component (tiles, links) within a selected time period is encoded with color. A darker color indicates more flits traversing the component. The traffic value associated with the darkest color is configurable, so users can set it to the capacity of each link and switch. This encoding supports identifying congested regions (**T2.2**), identifying components that generate messages (**T4.1**), and identifying message flow patterns (**T1.1.2**, **T3.2**) through spatial trends of color.

We label each link with the exact number of flits delivered in the selected time range. This allows computer architects to perform detailed evaluations or compare the flit numbers (e.g., 498 flits sent over a capacity of 500, one link serving 30 flits fewer than its neighbor).

It is not feasible to show the entire network for large-scale NOC systems. Thus, we enable visualization of larger NOCs (**DR4**) with two features: 1) a “Bird’s-Eye View” that always presents an overview of the whole network and 2) a semantic zooming feature for the detail view.

Bird’s-Eye View. The Bird’s-Eye View is a mini-map of the entire network overlaid at the bottom-left corner of the Detail View. Switches are combined into pixels, and links are elided to summarize traffic in regions (**T2.2**) and high-level patterns (**T1.1.2**, **T3.2**), allowing users to quickly identify these features across the network, thereby also helping to identify and characterize phases during initial overview (**T1.1**, **T1.2**).

An interactively linked brush over the Bird’s-Eye View shows the region displayed by the Detail View with respect to the entire mesh and can be used for quick navigation.

Semantic Zooming. The Detail View supports zooming and panning. As the scale of the network grows large, there is insufficient screen area to zoom out to the entire network. Thus, to ensure the Detail View is still comprehensible at high zoom out, we aggregate tiles and their associated links when zooming out as shown in Figure 4. Incoming and outgoing links are aggregated into arrows attached to next-level tiles. We label each aggregated tile with the true number of tiles it represents (e.g., 1x1, 4x4, 16x16, etc.) so users know the level they are examining. To avoid confusion caused by sudden aggregation level changes, we introduce an intermediate view (Figure 4 (b)) that shows both the low-level and next-level tiles. As a user zooms through aggregation levels, the tiles gradually shift through opaque and nested depictions. We chose to aggregate 4x4 tiles at each level to balance visibility (sufficient tiles on the screen) and rendering performance (avoid too many tiles on the screen).

6.2 Visualizing Temporal Traffic Distribution

While the Detail View and the Bird’s-Eye View provide spatial context to the data, the Temporal View (Figure 1 ①, the Temporal View) provides both summary and detail regarding the temporal context of the data. The Temporal View also serves as an important interactive control for faceting the data in time for the network views. Overall, the Temporal View uses the x-axis to represent time. We lay out two charts, the total traffic chart and the peak link-utilization chart, on the top and bottom of the time axis, respectively.

View Design. The Temporal View is designed to fulfill **DR2**, **T1.1** and **T2.1** and guide [10] users to select time ranges of interest.

Total traffic is shown as stacked bars, with the total bar height representing the number of flits transferred in the time slice across the entire network. Each color represents the number of flits that belongs to a type of traffic (e.g., reads, writes). We support four types of traffic that represent the commonly seen data types in GPU computing (**T1.1.1**). The classification can be configured using our data collection library described in subsection 5.4.

The total traffic bars help identify temporal patterns, such as phases in traffic (e.g., perhaps due to application behavior) (**T1.1**). For low-traffic regions, users can then examine architectural reasons why the network is underutilized (e.g., some component is not fast enough to generate sufficient traffic) as described in subsection 6.4. For high-traffic regions, users can determine if the network provides sufficient support for the architecture and the application with the spatial views.

Peak traffic is shown with a heat strip, where the color represents the number of flits that the most congested link transfers in the time slice. During our development, we observed that the traffic bars could be misleading as users assume low-traffic periods are congestion-free. However, it is common for users to miss that a few congested links may block the entire network, leading to low message counts despite high congestion. The heat strip shows cases of high saturation (**T2.1.2**) regardless of overall traffic. Users can determine the duration of high congestion in the network and further target their analysis.

Interaction design. Users can brush across the temporal view to filter the network views to specific time steps. Hovering over a time slice will reveal the exact value of the corresponding time in the simulation.

Animation. Vis4Mesh supports animation during which the brush automatically steps through time slices, updating the network view as the selected time slice changes. We design this feature to support both **DR1** and **DR2** (dependencies with time and network) as well as **T1** (identifying and characterizing phases).

6.3 Traffic Filters

In addition to the temporal filtering available through brushing the temporal panel, Vis4Mesh also has several filters for narrowing down the type of traffic shown in the rest of the visualization (**T1.1.1**, **T3.1**). These are based on attribute data associated with the flits.

Message Type Filter. We classify the flits by the high-level message types. The types used here are the same as those used in the stacked bars. These classes help distinguish the most common behaviors on the network and the impact they may have on resources. By turning any type on or off and observing the Detail view, users can quickly identify if a network behavior is caused by a particular architecture behavior (e.g., long-distance message passing is caused by address translation).

Transfer Type Filter. Another useful way to categorize flits is by their origin from the point of view of switches. The flit can originate at a component (e.g., an adapter), be relayed by a switch, be consumed by a component (e.g., the receiving adapter), or may come from a peripheral component outside the NOC system. We refer to these cases as the flit's transfer type as it traverses a component. For origination and consumption, we use the networking-specific abbreviations TX (transmit) and RX (receive), respectively.

Transfer type filtering helps users identify components that produce (**T4.2**) or consume many flits or identify when traffic is coming mostly from outside the network. They provide additional insight to message flow patterns (**T1.1.2**, **T3.2**).

We identified the utility of understanding transfer types through analysis sessions with the entire design team. The ChipTeam then added transfer type to flit data collection. We discuss the integration of data collection and visualization design further with the next filter.

Hop Filter. Analysts may be interested in messages (flits) that have long (or short) paths to reach their destination (**T3**). Thus, we enable users to filter traffic by the number of "hops" (transfers) a flit requires to reach its destination. This filter allows users to examine different kinds of traffic by distance and understand the proportion of long or short routes.

This hop-filtering feature was proposed during a group analysis session with the entire design team. To support the desired distance-based analysis, the ChipTeam then added a feature to append a hop-distance attribute to flits in the trace. In implementing both the data collection and data visualization requirements for hop-filtering, the ChipTeam determined that collecting exact hop counts per flit would greatly increase the data size. The ChipTeam thus decided to collect hop-count ranges as a trade-off between specificity and data size.

6.4 Architecture Drill-Down via Daisen Integration

Computer architects designing NOC systems require both the network context, which tracks data flows through the network, and the architecture context, which considers the architecture choices for the individual components using the network. The views we have presented thus far mainly focus on the network context.

Rather than designing new architecture context views, we chose to integrate an existing validated visualization tool, Daisen, for drilling down into architectural details (**T4**). This choice reduces design and development efforts and leverages prior knowledge for users already familiar with Daisen.

Daisen provides visual support for analyzing the detailed behavior of each individual component. The component view of Daisen can show all the tasks (e.g., instruction execution, memory access, flit transfer) completed by a component. Moreover, The task view of Daisen allows users to trace the tasks up (what task triggers the current task) and down (what subtasks need to be completed) the task hierarchy across components.

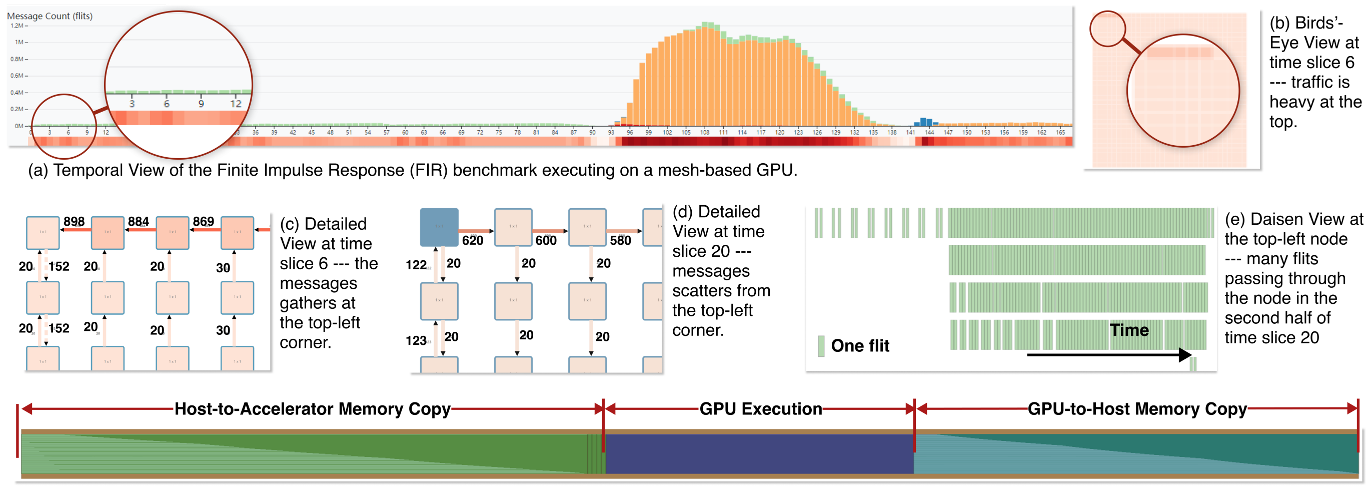
To support a tight analysis loop between the network and architecture contexts, we implemented coordination between Daisen and the Vis4Mesh views. Changes to the switch and time selection in the Vis4Mesh original views will update the Daisen views. Temporal zooms in Daisen will update the temporal panel brush in Vis4Mesh. If other components are associated with tasks selected by navigating Daisen, the Vis4Mesh Spatial View will re-center accordingly. These bi-directional synchronization features required small modifications to the Daisen code base.

7 Case Study

The ChipTeam conducted a case study analyzing the execution of the Finite Impulse Response (FIR) Filtering benchmark from the Hetero-Mark suite [57] simulated on a large-scale hypothetical GPU design. The GPU is organized in a 32 $\times$ 32 mesh. Each tile equips one Compute Unit (i.e., a GPU core) and a 4 MB Static Random Access Memory (SRAM) that stores the data that needs to be processed by the GPU. All data is initially located outside the GPU and needs to be copied to the GPU. After the program execution, the output data will be moved outside the GPU. The benchmark and the hypothetical hardware configuration originated from another research project in the ChipExpert's group.

Upon opening our simulation trace in Vis4Mesh, we first notice three distinct phases in the Temporal View (see Figure 5 (a)): two long periods with low traffic volume separated by a short phase with high traffic volumes. The heat strip along the bottom shows that while traffic volumes are low in the first and third phases, there are still individual links with medium traffic. Both high-traffic and low-traffic periods may need further investigation because both may have performance improvement opportunities. Since these low-traffic phases take up most of the time, we focus on the first such phase for our case study.

Animated visualization of the host-to-accelerator copy phase identifies periods of heavy network traffic in the top left quadrant. To examine these in more detail, we select a time slice for deeper



(a) Temporal View of the Finite Impulse Response (FIR) benchmark executing on a mesh-based GPU.

(b) Birds'-Eye View at time slice 6 --- traffic is heavy at the top.

(c) Detailed View at time slice 6 --- the messages gathers at the top-left corner.

(d) Detailed View at time slice 20 --- messages scatters from the top-left corner.

(e) Daisen View at the top-left node --- many flits passing through the node in the second half of time slice 20

(f) Daisen View of the whole program execution at the top-level of hierarchy. It explains what the system is working on different phases

Fig. 5. Major views used in the case study, analyzing the FIR benchmark running on a mesh-based GPU architecture.

analysis. From Figure 5 (b), we observe that the data-transfer hotspot occurs in the top left region.

To further investigate, we can use the Detail View to take a closer look at the exact number of flits sent over the top-left region links (see Figure 5 (c) and (d)). Using the animation and observing the numbers on the links, we see a clear gather and scatter pattern where the top-left node serves as a central hub. This pattern results in an unbalanced and underutilized mesh network, with data congested in a single node. Furthermore, by toggling the Hops Filter (results not shown due to limited space), we notice a large amount of traffic travels long distances, causing long delays.

At this point, we understand the traffic properties and have identified the congestion point. However, without linking the network behavior with the architectural behavior, it is difficult to understand why the messages must be sent and, therefore, to design an improved solution.

To examine how architectural behavior affects performance, we select the top-left node and open the Daisen panel to examine all the flits that pass through the node. With Daisen, we first observe that a large number of flits pass through the node (see Figure 5 (e)), which matches our observation in the Detail View. The question we want to answer is why there are so many flits at this time and location. We thus utilize Daisen to trace back the architectural tasks toward the root of the task tree.

As we trace back to the whole network level, we see that each flit traverses many hops (not shown due to space limit), confirming our observation with hop filters. Moreover, as we keep tracing until the Direct Memory Access (DMA) unit level, we see that the DMA unit initiates the traffic (also not shown due to space limit). Finally, at the whole benchmark level, we observe the timeline of the 3-stage application execution. The timeline in Daisen (Figure 5 (f)) matches with the timeline in Vis4Mesh (Figure 5 (a)), which can help users map the traffic phases with the program execution phases, enhancing their confidence in analysis results.

The observation at the DMA unit and the whole benchmark level suggests that the first low-traffic phase involves communication between elements outside and inside the mesh, passing through the top-left node. As the single entry point of the whole mesh, the top-left node suffers from high traffic pressure. This observation confirms that the long low-traffic phase is due to moving memory from outside the GPU. We can now consider designing alternatives either within the architecture or in how the NOC ingests data.

8 User Evaluation

To further validate the Vis4Mesh workflow and visual design, we conducted a preliminary qualitative user study with 7 GPU researchers and professionals. All sessions were conducted via online meeting software (i.e., Zoom) and recorded and transcribed. Two authors reviewed the recordings and transcripts, took notes, and open-coded the notes to find themes. This research was approved by our Institutional Review Board (IRB).

8.1 Method

We began each session with a demonstration using the example described in our case study (FIR Benchmark, section 7). Participants were then asked to perform a list of tasks based on a more complex dataset: the execution of a convolutional layer in a Convolutional Neural Network. The first task was an open exploration. Afterwards, we gave them shorter tasks that guided them through the workflow and asked them to identify features. Finally, we asked them about the overall bottleneck.

While performing tasks, participants were asked to think aloud and share their screens, allowing us to observe and understand their interactions. They were also given a PDF "cheatsheet" user guide for Vis4Mesh.

Following the tasks, we conducted semi-structured interviews to gather additional feedback. Finally, participants completed a survey regarding Vis4Mesh. We also used this survey to collect demographic information. Each session lasted between 75 and 90 minutes. We provide the tasks, interview questions, and survey in the supplemental materials.

Participants. We recruited seven participants with computer architecture experience through our networks. Table 8.1 summarizes their years of experience in architecture and their previous experience with Daisen. One is a Ph.D. student who works with ChipExpert. None are authors. The participants were not compensated for participating.

8.2 User Study Findings

The two ChipTeam students conducted the user study sessions. They discussed each session after the interview, going over the

TABLE 2
Participant demographics.

Participant	P1	P2	P3	P4	P5	P6	P7
Highest Edu.	B.S	M.S.	Ph.D.	B.S.	Ph.D.	Ph.D.	M.S.
Years Exp.	1-3	1-3	1-3	1-3	1-3	5-8	3-5
Daisen Exp.	No	No	No	Yes	Little	Little	Yes

recordings and transcripts to gather data relevant to the research questions and any other Vis4Mesh feedback. We present our findings below.

We did not expect participants to be familiar with Vis4Mesh. To keep the study a feasible length while still covering the major features of Vis4Mesh, the short tasks were designed to guide participants to use each feature along the expected workflow. Five of the seven participants followed the guide closely, while two preferred to explore it on their own first. Thus, our findings mostly pertain to feedback on the workflow and visual design. We cannot draw strong conclusions regarding how people would use Vis4Mesh if given just the tool.

Participants gained performance insights throughout our workflow from temporal overview to spatial and architectural detail. All participants identified bottlenecks during the session, but required additional help regarding architectural root causes in the final part of the workflow.

Participants were able to use the temporal panel to identify phases and execution patterns. P6 stated that "when I find patterns in the application, I feel that this is the most relevant. My analysis would start from this view first."

Insights from the stacked bars included speculations about architecture behavior such as memory copying, kernel launching and intensive computing. Additionally, some participants used the heat strip to identify bandwidth issues. P3 said the heat strip was the second most important feature because "we can detect when the traffic is bound by bandwidth and it shows us the bandwidth bottlenecks."

After filtering in time, Participants considered the spatial behavior of traffic with the bird's eye view. P1 noted "This is a very important feature and most of the time are spent on this bird's eye view." P7 said "During the early stage of network analysis, we can directly see the approximate location of the bottleneck through the bird's eye view."

Five of the seven participants also used the the transfer type and message type filters, but not the hop filter, to further refine both the temporal and spatial views. The traffic type was used to check sources and destinations, allowing them to further hypothesize about traffic patterns, such as message broadcasting. The message type filter hinted at architectural causes (e.g., memory reads and writes). P6 said the filters were "the most power" feature, noting "you can actually filter outside messages and just keep one type of message to see what the traffic is about."

Once a spatial location was found in the bird's eye view, several participants then successfully zoomed on the detailed view, though this was not as universally used as the other view. P2 said "I can zoom in layer-by-layer to see the details I want, without getting lost in too many details at first."

After identifying a particular component of interest in the spatial view, participants opened Daisen for more details on architectural behavior. They used Daisen to trace through processes within the component. P7 said "The integration with Daisen is very useful. You can actually find the bottleneck in architecture with Daisen, such as message latency and task duration." P2 noted Daisen provides an "instruction view" which "enables users to analyze the reason of a hot spot at the instruction level."

Participants gave positive feedback regarding our workflow. P1 commented "Vis4Mesh can provide information about when, where, and what the hotspot is and then Daisen tells why there is a hotspot." And P6 said "The workflow is very straightforward and intuitive. You have information of different granularity, and it is easy to target hotspot step by step".

Participants suggested additional refinement of zooming features. Though participants used the zooming in the detail view to locate components of interest and verify traffic patterns from the bird's eye view, some struggled with the semantic design. P1 suggested more explicit prompts were needed to help users understand the zoom level.

Participants also requested zooming features in the temporal panel. For example, P1 commented "It would be much better if Vis4Mesh supported zooming in on the temporal panel to focus on a specific time range." While participants were supportive of the animation feature, they did not utter insights during its use and one suggested animation might be better for replay in a smaller time range of the simulation.

Participants suggested improvements for the integration with Daisen. Three participants found Daisen tricky and less intuitive than Vis4Mesh. They recommended adding more tooltips and help materials. One participant (P1) also noted that Daisen increases the tool dependencies because it uses a back-end server, while otherwise Vis4Mesh could be run in a more portable fashion.

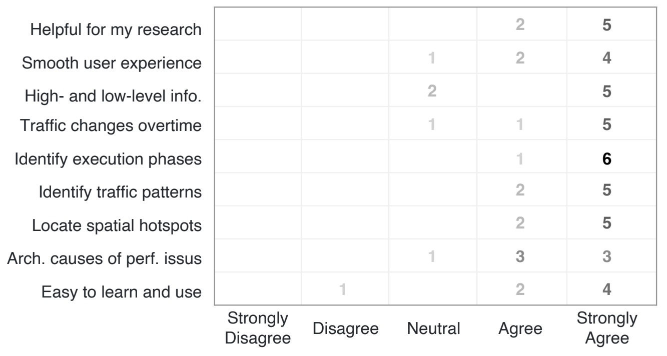
Integrating two tools provided many features, but also introduced new problems. Participants used Daisen to explore architectural causes for bottlenecks discovered in Vis4Mesh. They noted the benefits, but also the mismatch in complexity. P2 commented "Daisen provides information at the instruction level and it helps me understand why the hotspot takes place by telling me lots of instructions are issued." P7 said: "Daisen has massive data and provides lots of views which makes me get lost easily. However, Vis4Mesh allows me to find potential bottlenecks in an intuitive way and then I can use Daisen to get what I want.".

Participants found Vis4Mesh unique among architecture visualization tools. P1 commented: "Vis4Mesh is friendly and intuitive for users. It focuses more on network traffic other than architecture." P7 had the same feeling as "Vis4Mesh is for external Network-on-Chip, while Daisen is for internal GPU architecture." Besides, P6 noted that Vis4Mesh not only provides you static results, as "It has a lot of like interesting features. They are very interactive and very responsive.".

Quantitative Survey Results. Figure 6 shows the Likert scale responses to our post-study survey. The full survey prompts can be found in the supplemental materials. Participants were largely positive regarding Vis4Mesh's ability to aid with the given tasks. However, there were lower average scores regarding user experience/ease of use concerns as well as architectural concerns and high and low-level information. We discuss this further in the Discussion section below.

8.3 Discussion on User Study Findings

Overall participants used Vis4Mesh as expected by our design, with all of the features being used by at least some of the participants and all leading to performance analysis hypothesizing and insights



	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
Helpful for my research				2	5
Smooth user experience			1	2	4
High- and low-level info.			2		5
Traffic changes overtime			1	1	5
Identify execution phases				1	6
Identify traffic patterns				2	5
Locate spatial hotspots				2	5
Arch. causes of perf. issus			1	3	3
Easy to learn and use		1		2	4

Fig. 6. Post-study quantitative survey results, with the questions sorted in the same order as the survey.

by those who used them with the exception of the Hop Filter and the animation in the time view.

The Hop Filter was implemented by the ChipTeam to provide more information regarding how far flits have to travel in the NOC. Several trade-offs were made in data collection as true paths would be expensive to collect. We hypothesize that the inference expected was not obvious to the participants which may have been compounded by lack of familiarity with the presented evaluation NOC design.

We note that the lowest average score regarded architectural causes of performance issues. We suspect the difficulty in using Daisen, especially among the new users, may have contributed to this score. Additionally, participants expressed unfamiliarity with the architectural design of the given scenario. In practice, they would use Vis4Mesh on an architecture they were designing.

The combination of the two tools is tricky for both users and developers. The separate tool requires its own learning curve as well as additional data collection and deployment. We developed our tracing capabilities such that Vis4Mesh and Daisen data can be collected simultaneously. We also had to design a protocol and modify both tools to support double-ended communication.

There were also a few participants who were neutral regarding Vis4Mesh's ability to show both high and low-level information. Their scores may reflect some of the zooming issues found in the qualitative feedback.

8.4 Limitations and Threats to Validity

As we recruited participants through our own networks, the participants may be biased toward giving positive feedback. We have tried to discuss observations as well as direct feedback and carefully analyze where feedback and observations were more reserved, even when still positive.

Our user study is limited to seven participants. We were conscientious in recruitment as expert time is valuable. Thus, when we began to observe similar feedback, we decided to stop recruiting both to respect to our colleagues' time and reserve participants for future evaluations of Vis4Mesh or other architecture visualizations.

Vis4Mesh was designed under the assumption of regular use by people familiar with both the designed GPU architecture and the benchmark used to collect data. We supplied the data in this user study, so participants lacked that assumed background knowledge. The choice to supply data was because not all participants were actively working on a NOC design and we wanted to decrease the burden on participants in terms of running new tools to collect data for those who were. Thus, the feedback we received was speculative on the part of the participants for their own scenarios.

9 Domain Expert-Led Visualization Design: Reflections and Recommendations

We discuss advantages, disadvantages, pitfalls, and recommendations for domain expert-led visualization design projects.

The domain experts were inclined to intuitively design without making tasks explicit, leading to interventions by the visualization expert. The members of the ChipTeam, especially the ChipExpert, had strong initial notions of what they wanted to see and what their high-level goals were. They were eager to implement views and features without cataloging or examining their implied tasks. The VisExpert was concerned these design and development practices could lead to prioritizing less-useful features or creating a design based on incorrect or vague assumptions that could be avoided with more careful examination.

The VisExpert performed two interventions in the design process to better elicit tasks. The first was arguing for a broader point of view—seeking tasks outside the ChipTeam. This led to the NOC literature survey (subsection 5.2) which in turn led to the decision to focus on meshes and the choice of prevalent metrics.

The second intervention by the VisExpert was ongoing throughout the project. During the full team's regular meetings, the VisExpert deliberately probed the statements of the ChipTeam to refine the group's understanding of tasks. These statements included design proposals, goal statements, and utterances while using the prototype to examine sample data. It was during one of these meetings that the VisExpert recognized the mismatch between the visualization of total flits and the task of identifying congestion, leading to the addition of the heat strip in the temporal view.

The ChipTeam had little experience with task elicitation and documentation and were unsure of its value, especially when they "knew" what they wanted to see. The VisExpert had difficulty compelling the need for task analysis through stories of other projects. There is no way of knowing if the ultimate design would have been different without the VisExpert's interventions as design decisions could have come about in other ways, such as test-driving the prototypes with data. With the expertise of the ChipTeam, the extended task elicitation and analysis advised by the VisExpert here may be more about risk mitigation than in other projects where the designers are not the intended audience.

The ChipTeam was responsive to the VisExpert's suggestions due to trust in the VisExpert. Improved ways to quickly communicate the need for task analysis to non-visualization experts may be necessary for other domain expert-led design projects. While the first intervention, the literature survey, was a significant undertaking by the ChipTeam, we note the second intervention, deliberate probing in regular meetings, can be easily integrated without much disruption of the domain team, and recommend the action by people in the VisExpert's role.

The project setup led to a fluid data collection, data processing, and visualization design loop. The ChipTeam refined the data collection process while designing the visualization, as seen with the hop-filter feature. The VisExpert observed that trade-off decisions regarding the difficulty and resources (including human, computational, and storage resources) were made fluidly and rapidly in comparison to other visualization designs projects [62], likely because the ChipTeam had implementation ownership of both the data collection and the visualization.

Similarly, code design decisions about data processing, such as building the summed-area tables for the Temporal View or the hierarchies for the semantic zooming of the network, could be made more consistent across the data and visualization pipeline. Though the visualization is designed to run separately from the data collection so it can be used with other simulators, the data processing is consistent across ChipTeam's code because there is no translation or hand-off of data from the domain experts to the visualization designers.

Developing the visualization tool within the domain also builds expertise to maintain it, but resource issues persist. Vis4Mesh is developed within ChipTeam's organization, ensuring that ChipTeam has the expertise and familiarity to maintain and evolve it. However, there are still issues of maintenance. On reflection, the ChipExpert expressed regret in not having it developed within Daisen directly but as a separate tool and tracing library. Not integrating the existing tool chain may hinder adoption by potential users outside his team. Further, merging the tools increases the maintenance work. Discussing plans for integration and maintenance early in the project can make trade-offs explicit and provide the opportunity to alleviate some of these issues. The VisExpert can bring their experience and other perspectives of the visualization community, such as Akbaba et al.'s findings on maintenance in design studies [1].

The ChipExpert's sincere interest in visualization, tools, and design was essential in their leading the project. The ChipExpert consistently affirmed the importance of tools, including visual ones, in conducting computer architecture research. They had experience in previous visualization projects, familiarity with design choices beyond the selection of chart type (e.g., encoding choices, multi-view concerns), and experience with visualization libraries such as D3js. This enabled them to guide the ChipTeam on visualization matters without the VisExpert and understand and trust recommendations made by the VisExpert.

Though the ChipExpert is a computer architecture researcher, they were also interested in the visualization research aspects. This interest influenced the project process, including the formality of the evaluation, the choice of a design study over a design project, and the ultimate decision to present the work in publication. The decision to approach the project as visualization research was made early in the process. We recommend other domain expert-led visualization design projects determine research goals, if any, early so such process and methodology decisions can be made.

The VisExpert acted to fill knowledge and methodology gaps of the domain team. The VisExpert served to extend the visualization knowledge of the ChipExpert in terms of the design process, methodology, iterating on visual designs, designing evaluations, and providing context for possible visualization research contributions. The process and methodology guidance of the VisExpert served as a bridge to understanding existing literature and may suggest that more accessible guidance could help interested domain experts, especially those with programming skills, better design visual tools.

Summary of recommendations. Based on these experiences and reflections, we make the following recommendations for domain expert-led design studies:

R1 The domain and visualization experts should devise a collaboration schedule where the domain expert feels comfortable in directing their team between meetings while the visualization expert can intervene with guidance before design decisions become too hardened.

R2 Domain experts will need additional guidance and reinforcement regarding methodological aspects new to them, like task analysis and generation of non-code design artifacts.

R3 Discussions about research contributions and maintenance should occur early and be revisited regularly.

R4 The team should embrace the ability to extend data collection as permitted by the domain expert-led format.

From these recommendations, we note there is a need to create and curate translational materials and examples of design methodology that are more accessible to domain experts, other than visualization research papers.

10 Conclusion

We presented a domain expert-led design study in which the initial design, day-to-day design decisions, and implementation were managed by the domain experts. Through this design study, we developed and validated Vis4Mesh, an interactive multi-visualization that aids Network-on-Chip design through high- and low-level views in time and space. By integrating Daisen, an existing tool, into Vis4Mesh, we enabled a more comprehensive and detailed exploration of the computer architecture perspective of the on-chip network. Participants in our user study were able to identify performance bottlenecks with Vis4Mesh and affirmed the workflow we designed. Despite this success, we also identified areas for further research in terms of better navigation support between overview and detail in large datasets.

Our design study demonstrates that this form of research can be successfully led, with the consultation of a visualization expert, by domain experts who have computing skills and visualization interests. We observed benefits to this approach in terms of potential for data design, intrinsic domain understanding, and potential for rapid development. However, we also observed hazards in understanding methodological needs such as task consideration and even in maintenance despite the domain home of the tool. Additional guidance, early and throughout, may help alleviate these issues. We also note key factors in the success of this study, including the consultation and intervention actions by the visualization expert and the commitment level of the domain expert.

Acknowledgment

We sincerely thank the anonymous reviewers for the constructive feedback. This work is supported by the National Science Foundation (NSF) under OAC-2246035, CNS-2234400, and CNS-2234401. The work is also supported in part by AMD.

References

[1] D. Akbaba, D. Lange, M. Correll, A. Lex, and M. Meyer. Troubling collaboration: Matters of care for visualization design study. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pp. 1–15, 2023. doi: 10.1145/3544548.3581168

[2] A. Ariel, W. W. Fung, A. E. Turner, and T. M. Aamodt. Visualizing complex dynamics in many-core accelerator architectures. In *2010 IEEE International Symposium on Performance Analysis of Systems & Software (ISPASS)*, pp. 164–174. IEEE, 2010. doi: 10.1109/ISPASS.2010.5452029

[3] G. Ascia, V. Catania, A. Mineo, S. Monteleone, M. Palesi, and D. Patti. Improving inference latency and energy of dnns through wireless enabled multi-chip-module-based architectures and model parameters compression. In *2020 14th IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, pp. 1–6, Sep. 2020. doi: 10.1109/NOCS50636.2020.9241714

[4] J. Bashir, C. Goodchild, and S. R. Sarangi. Seconet: A security framework for a photonic network-on-chip. In *2020 14th IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, pp. 1–8, Sep. 2020. doi: 10.1109/NOCS50636.2020.9241713

[5] S. Bharadwaj, S. Das, Y. Eckert, M. Oskin, and T. Krishna. Dub: Dynamic underclocking and bypassing in nocs for heterogeneous gpu workloads. In *Proceedings of the 15th IEEE/ACM International Symposium on Networks-on-Chip*, NOCS '21, p. 49–54. Association for Computing Machinery, New York, NY, USA, 2021. doi: 10.1145/3479876.3481590

[6] A. Bhatele, N. Jain, Y. Livnat, V. Pascucci, and P.-T. Bremer. Analyzing network health and congestion in dragonfly-based supercomputers. In *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 93–102. IEEE, 2016. doi: 10.1109/IPDPS.2016.123

[7] B. Bisht and S. Das. Bht-noc: Blaming hardware trojans in noc routers. *IEEE Design & Test*, 39(6):39–47, Dec 2022. doi: 10.1109/MDAT.2022.3202998

[8] M. Bostock. D3.js - data-driven documents, 2012.

[9] K. A. Brown, J. Domke, and S. Matsuoka. Tracing data movements within MPI collectives. In *Proceedings of the 21st European MPI Users' Group Meeting*, pp. 117–118, 2014. doi: 10.1145/2642769.2642789

[10] D. Ceneda, T. Gschwandtner, T. May, S. Miksch, H.-J. Schulz, M. Streit, and C. Tominski. Characterizing guidance in visual analytics. *IEEE transactions on visualization and computer graphics*, 23(1):111–120, 2016. doi: 10.1109/TVCG.2016.2598468

[11] C. Chen, Z. Tao, and J. S. Miguel. Bufferless NoCs with scheduled deflection routing. In *2020 14th IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, pp. 1–6, Sep. 2020. doi: 10.1109/NOCS50636.2020.9241585

[12] Z. Chen, R. Deng, K. Zeng, X. Ni, and H. Zhou. Traversal packets: Opportunistic bypass packets for deadlock recovery. *IEEE Design & Test*, 39(6):48–57, Dec 2022. doi: 10.1109/MDAT.2022.3204201

[13] S. Cheng, P. De, S. H.-C. Jiang, and K. Mueller. *torusvis*ND: Unraveling high-dimensional torus networks for network traffic visualizations. In *2014 First Workshop on Visual Performance Analysis*, pp. 9–16. IEEE, 2014. doi: 10.1109/VPA.2014.7

[14] S. Devkota, P. Aschwanden, A. Kunen, M. Legendre, and K. E. Isaacs. CcNav: Understanding compiler optimizations in binary code. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):667–677, 2020. doi: 10.1109/TVCG.2020.3030357

[15] T. Fujiwara, P. Malakar, K. Reda, V. Vishwanath, M. E. Papka, and K.-L. Ma. A visual analytics system for optimizing communications in massively parallel applications. In *2017 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 59–70. IEEE, 2017. doi: 10.1109/VAST.2017.8585646

[16] V. Gogte, D. Lee, R. Parikh, and V. Bertacco. NoCVision: A network-on-chip dynamic visualization solution. In *Proceedings of the 8th International Workshop on Network on Chip Architectures*, pp. 21–26, 2015. doi: 10.1145/2835512.2835518

[17] J. A. González, D. Stathis, and A. Hemani. Synthesis of predictable global NoC by abutment in synchoros VLSI design. In *Proceedings of the 15th IEEE/ACM International Symposium on Networks-on-Chip*, NOCS '21, p. 61–66. Association for Computing Machinery, New York, NY, USA, 2021. doi: 10.1145/3479876.3481594

[18] M. D. Grammatikakis, V. Piperaki, and A. Papagrigoriou. Multilayer NoC firewall services: Case-study on e-health. In *Proceedings of the 15th IEEE/ACM International Symposium on Networks-on-Chip*, NOCS '21, p. 75–81. Association for Computing Machinery, New York, NY, USA, 2021. doi: 10.1145/3479876.3481598

[19] K. W. Hall, A. J. Bradley, U. Hinrichs, S. Huron, J. Wood, C. Collins, and S. Carpendale. Design by immersion: A transdisciplinary approach to problem-driven visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):109–118, 2020. doi: 10.1109/TVCG.2019.2934790

[20] R. Haynes, P. Crossno, and E. Russell. A visualization tool for analyzing cluster performance data. In *Third IEEE International Conference on Cluster Computing (CLUSTER'01)*, pp. 295–295. IEEE Computer Society, 2001. doi: 10.1109/CLUSTR.2001.959990

[21] K. E. Isaacs, A. G. Landge, T. Gamblin, P.-T. Bremer, V. Pascucci, and B. Hamann. Exploring performance data with Boxfish. In *2012 SC Companion: High Performance Computing, Networking Storage and Analysis*, pp. 1380–1381. IEEE, 2012. doi: 10.1109/SC.Companion.2012.202

[22] J. M. Joseph, M. S. Baloglu, Y. Pan, R. Leupers, and L. Bamberg. NEWROMAP: Mapping CNNs to NoC-interconnected self-contained data-flow accelerators for edge-AI. In *Proceedings of the 15th IEEE/ACM International Symposium on Networks-on-Chip*, NOCS '21, p. 15–20. Association for Computing Machinery, New York, NY, USA, 2021. doi: 10.1145/3479876.3481591

[23] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, pp. 1–12, 2017. doi: 10.1145/3079856.3080246

[24] D. C. Jung, S. Davidson, C. Zhao, D. Richmond, and M. B. Taylor. Ruche networks: Wire-maximal, no-fuss NoCs : Special session paper. In *2020 14th IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, pp. 1–8, Sep. 2020. doi: 10.1109/NOCS50636.2020.9241586

[25] E. Kerzner, S. Goodwin, J. Dykes, S. Jones, and M. Meyer. A framework for creative visualization-opportunities workshops. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):748–758, 2019. doi: 10.1109/TVCG.2018.2865241

[26] L. C. Koh, A. Slingsby, J. Dykes, and T. S. Kam. Developing and applying a user-centered model for the design and implementation of information visualization tools. In *2011 15th International Conference on Information Visualisation*, pp. 90–95. IEEE, 2011. doi: 10.1109/IV.2011.32

[27] H. Krichene and J.-M. Philippe. Analysis of on-chip communication properties in accelerator architectures for deep neural networks. In *Proceedings of the 15th IEEE/ACM International Symposium on Networks-on-Chip*, NOCS '21, p. 9–14. Association for Computing Machinery, New York, NY, USA, 2021. doi: 10.1145/3479876.3481588

[28] V. J. Kulkarni, R. Manju, R. Gupta, J. Jose, and S. Nandi. Packet header attack by hardware trojan in NoC based TCMP and its impact analysis. In *Proceedings of the 15th IEEE/ACM International Symposium on Networks-on-Chip*, NOCS '21, p. 21–28. Association for Computing Machinery, New York, NY, USA, 2021. doi: 10.1145/3479876.3481597

[29] A. G. Landge, J. A. Levine, A. Bhatele, K. E. Isaacs, T. Gamblin, M. Schulz, S. H. Langer, P.-T. Bremer, and V. Pascucci. Visualizing network traffic to understand the performance of massively parallel simulations. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2467–2476, 2012. doi: 10.1109/TVCG.2012.286

[30] J. Landstorfer, I. Herrmann, J.-E. Stange, M. Dörk, and R. Wettach. Weaving a carpet from log entries: A network security visualization built with co-creation. In *2014 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 73–82. IEEE, 2014. doi: 10.13140/2.1.1116.3843

[31] I. Lang, N. Kapre, and R. Pellizzoni. Worst-case latency analysis for the versal NoC network packet switch. In *Proceedings of the 15th IEEE/ACM International Symposium on Networks-on-Chip*, NOCS '21, p. 55–60. Association for Computing Machinery, New York, NY, USA, 2021. doi: 10.1145/3479876.3481593

[32] N. Leyva, A. Monemi, and E. Vallejo. SynFull-RTL: Evaluation methodology for RTL NoC designs. *IEEE Design & Test*, 39(6):58–69, Dec 2022. doi: 10.1109/MDAT.2022.3202996

[33] Y. Li, M. Wu, W. Li, R. Xue, D. Fan, D. Li, Y. Ji, and X. Ye. An efficient multicast router using shared-buffer with packet merging for dataflow architecture. In *2020 14th IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, pp. 1–8, Sep. 2020. doi: 10.1109/NOCS50636.2020.9241709

[34] Z. Li, Y. Ma, G. Du, X. Wang, Y. Song, and D. Zhang. RB-OLITS: A worst case reorder buffer size reduction approach for 3-D-NoC. *IEEE Design & Test*, 39(6):79–89, Dec 2022. doi: 10.1109/MDAT.2022.3202993

[35] H. Luan and A. Gatherer. Combinatorics and geometry for the many-ported, distributed and shared memory architecture. In *2020 14th IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, pp. 1–6, Sep. 2020. doi: 10.1109/NOCS50636.2020.9241708

[36] H. Luan, Y. Yao, and C. Huang. A many-ported and shared memory architecture for high-performance ADAS SoCs. *IEEE Design & Test*, 39(6):5–15, Dec 2022. doi: 10.1109/MDAT.2022.3202997

[37] U. Mallappa, C.-K. Cheng, and B. Lin. JARVA: Joint application-aware oblivious routing and static virtual channel allocation. *IEEE Design & Test*, 39(6):16–27, Dec 2022. doi: 10.1109/MDAT.2022.3202994

[38] R. Manju, A. Das, J. Jose, and P. Mishra. SECTAR: Secure NoC using trojan aware routing. In *2020 14th IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, pp. 1–8, 2020. doi: 10.1109/NOCS50636.2020.9241711

[39] C. M. McCarthy, K. E. Isaacs, A. Bhatele, P.-T. Bremer, and B. Hamann. Visualizing the five-dimensional torus network of the IBM Blue Gene/Q. In *2014 First Workshop on Visual Performance Analysis*, pp. 24–27. IEEE, 2014. doi: 10.1109/VPA.2014.10

[40] S. McKenna, D. Staheli, C. Fulcher, and M. Meyer. BubbleNet: A cyber security dashboard for visualizing patterns. *Computer Graphics Forum*, 35(3):281–290, 2016. doi: 10.1111/cgf.12904

[41] G. Molina León and A. Breiter. Co-creating visualizations: A first evaluation with social science researchers. *Computer Graphics Forum*, 2020. doi: 10.1111/cgf.13981

[42] L. Möller, L. S. Indrusiak, and M. Glesner. NoCScope: A graphical interface to improve networks-on-chip monitoring and design space

exploration. In *2009 4th International Design and Test Workshop (IDT)*, pp. 1–6. IEEE, 2009. doi: 10.1109/IDT.2009.5404105

[43] A. Monemi, I. Pérez, N. Leyva, E. Vallejo, R. Beivide, and M. Moretó. PIugSMART: A pluggable open-source module to implement multihop bypass in networks-on-chip. In *Proceedings of the 15th IEEE/ACM International Symposium on Networks-on-Chip*, NOCS '21, p. 41–48. Association for Computing Machinery, New York, NY, USA, 2021. doi: 10.1145/3479876.3481601

[44] F. Muñoz Martínez, J. L. Abellán, M. E. Acacio, and T. Krishna. A novel network fabric for efficient spatio-temporal reduction in flexible dnn accelerators. In *Proceedings of the 15th IEEE/ACM International Symposium on Networks-on-Chip*, NOCS '21, p. 1–8. Association for Computing Machinery, New York, NY, USA, 2021. doi: 10.1145/3479876.3481602

[45] Y. Ou, S. Agwa, and C. Batten. Implementing low-diameter on-chip networks for manycore processors using a tiled physical design methodology. In *2020 14th IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, pp. 1–8, Sep. 2020. doi: 10.1109/NOCS50636.2020.9241710

[46] D. Petrisko, C. Zhao, S. Davidson, P. Gao, D. Richmond, and M. B. Taylor. NoC symbiosis (special session paper). In *2020 14th IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, pp. 1–8, Sep. 2020. doi: 10.1109/NOCS50636.2020.9241584

[47] A. Psistakis, N. Chrysos, F. Chaix, M. Asiminakis, M. Gianníoudis, P. Xirouchakis, V. Papaefstathiou, and M. Katevenis. PART: Pinning avoidance in rdma technologies. In *2020 14th IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, pp. 1–8, Sep. 2020. doi: 10.1109/NOCS50636.2020.9241587

[48] A. Sarihi, A. Patooghy, M. Hasanzadeh, M. Abdelrehim, and A.-H. A. Badawy. Securing network-on-chips via novel anonymous routing. In *Proceedings of the 15th IEEE/ACM International Symposium on Networks-on-Chip*, NOCS '21, p. 29–34. Association for Computing Machinery, New York, NY, USA, 2021. doi: 10.1145/3479876.3481592

[49] M. Sedlmair, M. Meyer, and T. Munzner. Design study methodology: Reflections from the trenches and the stacks. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2431–2440, 2012. doi: 10.1109/TVCG.2012.213

[50] A. Shalaby, Y. Tavva, T. E. Carlson, and L.-S. Peh. Sentry-NoC: A statically-scheduled NoC for secure SoCs. In *Proceedings of the 15th IEEE/ACM International Symposium on Networks-on-Chip*, NOCS '21, p. 67–74. Association for Computing Machinery, New York, NY, USA, 2021. doi: 10.1145/3479876.3481595

[51] S. Simon, S. Mittelstädt, D. A. Keim, and M. Sedlmair. Bridging the gap of domain and visualization experts with a liaison. In E. Bertini, J. Kennedy, and E. Puppo, eds., *Eurographics Conference on Visualization (EuroVis) - Short Papers*. The Eurographics Association, 2015. doi: 10.2312/eurovisshort.20151137

[52] V. Sobhani, K. Kauth, T. Stadtmann, and T. Gemmeke. Deadlock-freedom in computational neuroscience simulators. *IEEE Design & Test*, 39(6):70–78, Dec 2022. doi: 10.1109/MDAT.2022.3204199

[53] C. Sudusinghe, S. Charles, S. Ahangama, and P. Mishra. Eavesdropping attack detection using machine learning in network-on-chip architectures. *IEEE Design & Test*, 39(6):28–38, Dec 2022. doi: 10.1109/MDAT.2022.3202995

[54] C. Sudusinghe, S. Charles, and P. Mishra. Denial-of-service attack detection using machine learning in network-on-chip architectures. In *Proceedings of the 15th IEEE/ACM International Symposium on Networks-on-Chip*, NOCS '21, p. 35–40. Association for Computing Machinery, New York, NY, USA, 2021. doi: 10.1145/3479876.3481589

[55] K. L. Summers, T. P. Caudell, K. Berkbigler, B. Bush, K. Davis, and S. Smith. Graph visualization for the analysis of the structure and dynamics of extreme-scale supercomputers. *Information Visualization*, 3(3):209–222, 2004. doi: 10.1057/palgrave.ivs.9500079

[56] Y. Sun, T. Baruah, S. A. Mojumder, S. Dong, X. Gong, S. Treadway, Y. Bao, S. Hance, C. McCardwell, V. Zhao, et al. MGPUSim: Enabling multi-GPU performance modeling and optimization. In *Proceedings of the 46th International Symposium on Computer Architecture*, pp. 197–209, 2019. doi: 10.1145/3307650.3322230

[57] Y. Sun, X. Gong, A. K. Ziabari, L. Yu, X. Li, S. Mukherjee, C. McCardwell, A. Villegas, and D. Kaeli. Hetero-mark, a benchmark suite for CPU-GPU collaborative computing. In *2016 IEEE International Symposium on Workload Characterization (IISWC)*, pp. 1–10. IEEE, 2016. doi: 10.1109/IISWC.2016.7581262

[58] Y. Sun, Y. Zhang, A. Mosallaei, M. D. Shah, C. Dunne, and D. Kaeli. Daisen: a framework for visualizing detailed GPU execution. In *Computer Graphics Forum*, vol. 40, pp. 239–250. Wiley Online Library, 2021. doi: 10.1111/cgf.14303

[59] L. Theisen, A. Shah, and F. Wolf. Down-to-earth: How to visualize traffic on high-dimensional torus networks. In *2014 First Workshop on Visual Performance Analysis*, pp. 17–23. IEEE, 2014. doi: 10.1109/VPA.2014.6

[60] S. S. Vatsavai, V. S. P. Karempudi, and I. Thakkar. PROTEUS: Rule-based self-adaptation in photonic NoCs for loss-aware co-management of laser power and performance. In *2020 14th IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, pp. 1–8, Sep. 2020. doi: 10.1109/NOCS50636.2020.9241712

[61] J. Wang, Y. Huang, M. Ebrahimi, L. Huang, Q. Li, A. Jantsch, and G. Li. VisualNoC: A visualization and evaluation environment for simulation and mapping. In *Proceedings of the Third ACM International Workshop on Many-core Embedded Systems*, pp. 18–25, 2016. doi: 10.1145/2934495.2949544

[62] K. Williams, A. Bigelow, and K. E. Isaacs. Visualizing a moving target: A design study on task parallel programs in the presence of evolving data and concerns. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):1118–1128, 2020. doi: 10.1109/TVCG.2019.2934285

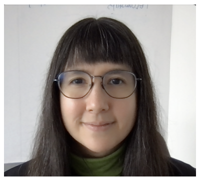
[63] J. Yin and A. Zhai. In-network memory access ordering for heterogeneous multicore systems. In *2020 14th IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, pp. 1–8, Sep. 2020. doi: 10.1109/NOCS50636.2020.9241583

[64] Y. Zhang, Y. Sun, J. D. Gaggiano, N. Kumar, C. Andris, and A. G. Parker. Visualization design practices in a crisis: Behind the scenes with COVID-19 dashboard creators. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):1037–1047, 2022. doi: 10.1109/TVCG.2022.3209493

[65] A. K. Ziabari, R. Ubal, D. Schaa, and D. Kaeli. Visualization of OpenCL application execution on CPU-GPU systems. In *Proceedings of the Workshop on Computer Architecture Education*, pp. 1–8, 2015. doi: 10.1145/2795122.2795125

Shaoyu Wang is a B.Eng. undergraduate student in Computer Science at Huazhong University of Science and Technology. His research focuses on systems for AI and computer architecture, including large-scale deep learning, domain-specific architecture, and software/hardware co-design.

Hang Yan is an undergraduate student pursuing a B.Eng. degree in Computer Science at Huazhong University of Science and Technology. His research interests broadly lie in computer architecture and high-performance computing, including data center heterogeneous computing, domain-specific architecture, and software/hardware co-design.



Katherine E. Isaacs is an Associate Professor in the Scientific Computing and Imaging (SCI) Institute and Kahlert School of Computing at the University of Utah. Her research focuses on data visualization challenges in complex exploratory analysis scenarios such as those of active research teams.

Yifan Sun is an Assistant Professor in Computer Science at William & Mary. He received his Ph.D. in Computer Engineering from Northeastern University in 2020. His interests include GPU architecture, performance evaluation, and performance modeling.