IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received August 30, 2020, accepted September 29, 2020, date of publication October 12, 2020, date of current version October 22, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3030606

NoC^2: An Efficient Interfacing Approach for Heavily-Communicating NoC-Based Systems

AHMED A. MORGAN[1,2], AHMED S. HASSAN[3], M. WATHEQ EL-KHARASHI[3,4], AND AYMAN TAWFIK[5], (Member, IEEE)
[1]Department of Computer Engineering, Faculty of Engineering, Cairo University, Giza 12613, Egypt
[2]College of Computer and Information Systems, Umm Al-Qura University, Makkah 21955, Saudi Arabia
[3]Computer and Systems Engineering Department, Faculty of Engineering, Ain Shams University, Cairo 11517, Egypt
[4]Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC V8W 3P6, Canada
[5]Electrical Engineering Department, Ajman University, Ajman, United Arab Emirates

Corresponding author: M. Watheq El-Kharashi (watheq@engr.uvic.ca)

ABSTRACT Current research in interfacing clusters within Hierarchical Networks-on-Chip (HNoC) as well as interfacing NoC-based systems adopts a centralized approach. In this approach, a specific Processing Element (PE) acts as a gateway between interfacing peripherals and the rest of NoC elements. This article evaluates this approach and show that it is not optimal for handling the inter-NoC communication. Routing inter-NoC traffic through a system to its gateway PE deteriorates the network performance. Results show that both the throughput and latency of the centralized approach degrade with the increase in the inter-NoC traffic bandwidth. To alleviate this, we propose a novel distributed approach, which separates the inter-NoC traffic from the intra-NoC one. Our approach relies on distributed buffers to allow PEs to efficiently communicate with the interfacing peripheral. We evaluate our approach against other interfacing ones using synthetic traffic as well as real benchmark applications. Our evaluation covers the whole system performance as well as its inter- and intra-NoC parts. Results prove that the proposed approach outperforms previous interfacing ones in terms of throughput and latency. The proposed approach significantly enhances the inter-NoC performance without any deterioration in the intra-NoC one. Considering the inter-NoC performance, we achieve a throughput that is close to the maximum possibly attainable one. Other approaches show major performance degradation, reaching as low as 10% of this maximum attainable throughput.

INDEX TERMS Hierarchical networks-on-chip (HNoC), inter-NoC communication, intra-NoC communication, NoC benchmarks, NoC Ethernet, NoC high-speed interfacing, NoC time division multiple access (TDMA), NoC traffic.

I. INTRODUCTION

Networks-on-Chip (NoCs) became widely used as a communication infrastructure for multicore and High Performance Computing (HPC) systems. Modern and future applications, which run on these systems, require a continuously increasing performance. From an NoC perspective, research efforts to fulfill this performance could be categorized into two main directions. In the first direction, the underlying NoC architecture is considered as a one consolidated system, such that the hardware resources are not grouped into clusters [1]. Numerous research work are presented to realize the required performance of applications running on these non-clustered architectures. Many static and dynamic techniques are proposed to customize the NoC hardware, according to the target application. For example, task-to-core mapping, routing, dynamic resource management, and other techniques are extensively researched. In this article, we are not contradicting with previous techniques in this direction or replacing them. Rather, we acknowledge that they successfully accomplish the performance requirements of many NoC applications. Nevertheless, as the number of cores and the communication among these cores increase rapidly in modern and future NoC-based systems, these techniques would struggle to fulfill the required performance of such complicated systems [2]. Managing and controlling the underlying hardware for a complex application, or multiple simultaneously running applications, would definitely be a challenge. Execution time, chip area, and power consumption budgets of these techniques would prevent many of them

The associate editor coordinating the review of this manuscript and approving it for publication was Nitin Nitin.

from being widely used with future heavily-communicating NoC-based systems.

The second research direction appears more promising and manageable for future NoC-based systems. In this direction, the underlying NoC architecture is partitioned into multiple clusters. In the literature, these Clustered NoCs (CNoCs) are also named Hierarchical NoCs (HNoCs) [3], or region-based NoCs [4]. Each cluster has a limited number of cores and could be easily managed and controlled. For a complex application, its tasks are also split into different segments. Each segment is mapped to one of the hardware clusters. Due to this task splitting, clusters are interfaced and connected to each others in order to carry out the whole application execution. Many techniques are presented to accomplish the clustering process and to customize the underlying NoC architecture to the target application. A common and key objective of these techniques is to minimize the inter-cluster traffic. This traffic minimization is intended to avoid congestion over the few inter-cluster links, to minimize the latency, and to maximize the throughput. In this article, we are not replacing these techniques and we again acknowledge the good performance they achieve. Nevertheless, most of these techniques reduce the inter-cluster traffic, but they don not eliminate it. Therefore, an open research challenge is how to properly handle this inter-cluster traffic. In other words, an efficient interfacing approach would indeed help previous techniques reaching the maximum attainable performance from the underlying hardware. On another hand, the future prediction of NoC-based systems shows a significant increase in their inter-cluster traffic [5]. This apparently emphasizes the need for the aforementioned efficient interfacing approach. However, in the literature, very few techniques are proposed to carry out the inter-cluster communication. In this article, we will show that the performance of these techniques is not optimal. Moreover, supported by our initial work in [6], we present a more efficient interfacing approach. Our approach is meant to work in conjunction with previous CNoC architecture customization techniques to properly handle their inter-cluster traffic and further enhance their performance.

Another method to realize the required increasing performance of future NoC-based systems would be through adding more computational cores. In other words, a new off-chip, or maybe off-board, NoC-based system would be connected to an already running one. Connecting an NoC-based system to another one necessitates an efficient interfacing approach to handle the communication among these systems. This interfacing approach should be taken into account during the design phase and it should further employ a standard communication protocol. Despite the importance of this challenge, it is barely studied by the NoC research community. In this article, we target this open research area. Our proposed interfacing approach could not only be used in connecting clusters within a single NoC-based system, but also in connecting different systems. In the later case, we are building a network of NoC-based systems. Therefore, we name our interfacing approach NoC^2. Finally, without loss of generality, our approach employs the Ethernet standard to carry out the communication among these systems.

The rest of the paper is organized as follows. Section II defines terms, which are frequently utilized throughout this article and clarifies the used terminology. Section III verifies the necessity for an efficient NoC interfacing approach. It highlights the prediction for the future inter-cluster traffic and summarizes the drawbacks with current interfacing approaches. Our contributions are then enumerated in Section IV. Section V reviews the related work. It first covers some research efforts to customize NoC architectures with minimal inter-cluster traffic. Thereafter, it surveys previous NoC interfacing techniques. Section VI presents our proposal, NoC^2, for interfacing NoC-based clusters, or systems.[1] Section VII explains the experimental setup employed in evaluating different NoC interfacing approaches. The section describes the used benchmark applications, the employed clustering variants, and the environment for simulation as well as hardware implementation. Section VIII shows and discusses results of evaluating different NoC interfacing approaches. Finally, the paper is concluded in Section IX.

II. DEFINITIONS AND USED TERMINOLOGY

Terms within the NoC research community might be used differently from one article to another. To avoid confusing the reader and make our discussion clear, we herein define terms that we are using throughout this article.

- **NoC architecture**: The actual hardware of the NoC-based system. In this article, we assume that the architecture consists of tiles, where each tile has the three following modules[2]
    1) Processing Element (PE), or core: The actual execution module, which runs a task, or tasks, of the application. In this article, PEs are drawn as black square Quad Flat Package (FQP) chips.
    2) Network Interface (NI): The module that interfaces the PE to the network. It packetizes the traffic at the source PE and de-packetizes the traffic at the destination PE. In our proposed approach, it is also responsible for communicating with the interfacing peripheral. In this article, NIs are drawn as while rectangles, with the abbreviation NI written inside these rectangles.
    3) Router: The module that executes the routing protocol. It transfers packets through the network from one tile to another. In this article, routers are drawn as circles with small arrows inside.
- **Non-clustered architecture**: The architecture, which is not partitioned into clusters. The system could simply be envisioned as a collection of tiles.

[1]For shortness, we will only use one of the two terms, either clusters or systems. Unless explicitly stated, the discussion is however applied for both.

[2]In our proposed approach, the communication module is also considered part of the architecture

- **Clustered or hierarchical architecture (CNoC or HNoC)**: The architecture, which is partitioned into clusters. Each cluster has multiple tiles. The system could be envisioned as a collection of clusters, which are interfaced to each others.
- **NoC interfacing**: Connecting clusters of a single system or connecting separate NoC-based systems through a standard communication protocol. Without loss of generality, Ethernet is used in this article.
- **Intra-NoC traffic**: The traffic exchanged among PEs of the same cluster, or the same non-clustered system.
- **Inter-NoC traffic**: The traffic exchanged among PEs of different clusters, or systems.
- **Benchmark**: A standard-alike NoC application, which is widely used by the NoC research community in evaluating the performance of novel proposed techniques.
- **Core graph**: For each benchmark, it is a graph, which represents the required PEs to execute the benchmark and the traffic exchanged among these PEs.
- **Useful throughput, or goodput**: The amount of traffic that is successfully delivered to its destination PE in a unit of time. Throughout this article, we drop the word "useful" and the term "throughput" is solely used to represent this goodput.
- **Total throughput, or router load**: The total amount of traffic passing through all ports of a router in a unit of time. This includes useful traffic as well as overheads. To avoid any confusion with the goodput, we use the term "router load" throughout this article to represent this total throughput.
- **Peak throughput**: The maximum throughput that could theoretically be achieved. It is calculated using a zero-delay model and assuming a congestion-free NoC with unlimited buffers. It represents a ceiling of the throughput in order to compare the performance of different interfacing approach against it.

III. NECESSITY OF EFFICIENT NoC INTERFACING APPROACHES

In this section, we aim to clearly explain the motivations behind our work. First, we discuss the expected increase in the inter-NoC traffic, which necessitates an efficient interfacing approach to handle it. Thereafter, we explore the drawbacks of current interfacing techniques, which encourage us to present a more efficient one.

A. PREDICTION OF FUTURE INTER-NoC TRAFFIC

NoC becomes the main communication backbone within modern multicore systems. The number of cores within these systems increases rapidly. Graphical Processing Unit (GPU) industry first led this trend by introducing cards with thousands of cores. Nickolls and Dally showed that the number of cores within modern GPUs increases according to Moore's rate, such that it doubles every 18 months [7]. In recent years, a significant increase in the number of cores also occurs in the embedded domains, general-purpose processors, and Application-Specific Integrated Circuits (ASICs) [8]. As mentioned in Section I, these cores are partitioned into clusters that are interfaced to each others. Consequently, the amount of inter-NoC traffic is significantly increasing and it should be properly handled.

Manevich *et al.* analyzed and modeled the inter-NoC traffic resulted from interfacing clusters within modern NoC-based systems [5],. Their work is based on a Rentian traffic model [9]. They further considered two modeling scenarios. The first is an optimistic one of lightly communicating clusters, whereas the second is a worst-case pessimistic scenario of clusters that exchange high traffic volumes. As the number of cores doubles, they predicted that the inter-NoC traffic would increase by 63.3% and 77.2% for lightly and heavily communicating clusters, respectively. Authors further signified that current interfacing techniques would soon suffer from network congestion and saturation, which directly affect the throughput, latency, and power consumption. Going with the aforementioned Moore's rate observation, the inter-NoC traffic would double approximately every 20.3 months, in the worst-case. Indeed, this is a significant increase in the inter-NoC traffic, which necessitates an efficient interfacing approach to handle it.

B. DRAWBACKS OF CURRENT INTERFACING TECHNIQUES

In the literature, few techniques are presented to carry out the interfacing process. Most of these techniques deploy the same approach. In this article, we call it the centralized approach. In this approach, only one centralized module, i.e., a PE or an NI, is used as a gateway between all cores in its system and the Communication Peripheral Controller (CPC) [10], [11]. An analogous approach is presented by Dorai *et al.* to alleviate the interfacing burden from PEs and NIs [12]. As such, no centralized gateway PE, or NI, is used and cores within the NoC-based system are allowed to access the CPC in a Time-Division Multiple Access (TDMA) fashion. While this approach is not centralized around the gateway PE, or NI, it is still centralized around the TDMA controller. In summary, the centralized approach suffers from four main drawbacks.

1) As being centralized, the gateway module becomes a bottleneck. As the traffic increases, this module constitutes a hotspot in the system, degrading the overall system performance.
2) The centralized approach requires that the inter-NoC traffic, from all cores within the system, be first routed to its gateway module. Thereafter, the traffic goes from this module to the interfaced system. Routing the inter-NoC traffic through the system rises the underlying network load, overwhelms its resources, increases its overall latency, and probably pushes it into congestion.
3) Once a new NoC-based system is to be interfaced using the centralized approach, the gateway module as well as its running software should be modified. This

makes the interfacing process more difficult and time consuming.

4) The centralized module might serialize the accesses to the CPC. For example, when using TDMA, some inter-NoC traffic would stall awaiting their designated time slots. This stall consequently increases the overall latency and affects the system performance badly.

IV. CONTRIBUTIONS

Motivated by the future prediction of inter-NoC traffic and the challenges of current interfacing techniques, we aim to present an efficient approach to handle the inter-NoC traffic and overcome the drawbacks of the centralized approach. Accordingly, our proposed NoC^2 approach eliminates the centralized gateway module in order to avoid creating hotspots in the system. It allows direct and concurrent access to the CPC in order to reduce the overall latency as well as congestion probability. It requires no modifications to the running software in order to make interfacing NoC-based systems much easier.

As our approach is proposed to efficiently handle the inter-NoC traffic, it mainly targets two important NoC metrics, the throughout and the latency. Consequently, the performance of our approach is evaluated against that of the non-clustered, the centralized, and the TDMA techniques, in terms of these two metrics. On another hand, associated overheads with the approach should be tolerable. In other words, our approach should be customized to minimize its impact on the NoC design budget. Buffers are shown to be responsible for most of the design area and power consumption [13]. Therefore, our approach employs as much buffers as those used by the centralized approach. Nevertheless, these buffers are re-distributed throughout the network more wisely.

Our approach is evaluated using simulation as well as hardware implementation on FPGA. The evaluation is conducted using synthetic traffic and real benchmark applications. For each employed benchmark, its cores are partitioned into two clusters. These clusters are then interfaced to each other. Thereafter, both inter- and intra-NoC communication performances are assessed, in terms of throughput and latency. As the performance of the overall system would be dependent on the used clustering technique, we consider two clustering variants. The first variant maximizes the inter-NoC traffic in order to judge the efficiency of different interfacing approaches, in the case of heavily-communicating systems. This variant would therefore evaluate whether a hotspot is created or a congestion is occurred. The second clustering variant yields the minimum inter-NoC traffic, and hence, the maximum intra-NoC one. This variant would consequently assess the effect of re-distributing NoC buffers on the intra-NoC performance. To this end our contributions are two fold:

1) Presenting a novel approach, NoC^2, for interfacing NoC-based clusters or systems. Our approach does not overload the network in terms of buffers. It further enhances inter-NoC communication performance without affecting the intra-NoC one. Once an NoC-based system is designed according to our approach, interfacing it to a similar system would not require modifying the cores nor the software running on them.
2) Evaluating different NoC interfacing approaches using synthetic traffic and real NoC benchmark applications through simulation as well as hardware implementation.

V. RELATED WORK

In this section, we review some of the related research efforts. First, in Subsection V-A, techniques that customize the NoC architecture with minimal inter-NoC traffic are surveyed. As mentioned in Section I, our proposed approach is to work in conjunction with these techniques to better enhance the performance of NoC-based systems. Thereafter, previous interfacing techniques are discussed in details in Subsection V-B. Those techniques are the ones which, we consider in our comparison in Section VIII to verify the efficiency of our proposed interfacing approach in handling the inter-NoC communication.

A. NoC CUSTOMIZATION TECHNIQUES WITH MINIMAL INTER-NoC TRAFFIC

Numerous techniques were presented to realize the best performance for NoC-based systems. Most of these techniques could be used with clustered and non-clustered architectures. A main objective of these techniques is to achieve a better traffic localization, such that the inter-NoC traffic is minimized [14]. As discussed in Section III, this traffic localization would not be easily achieved with the continuous increase in the number of cores and the traffic of future NoC-based systems. In summary, the most widely used NoC customization techniques are optimal task-to-core mapping, dynamic architecture reconfiguration, adaptive routing, traffic shaping and regulation, virtual channel partitioning, dynamic resource management, and network coding. Multiple of these techniques could be combined to achieve better performance. In the following paragraphs, we shed some light on these techniques.

Optimizing the task-to-core mapping is probably the most widely used technique for enhancing the performance of NoC-based systems. In this technique, mapping the application tasks onto the architecture cores is optimized for a certain metric, such as latency, throughput, power consumption, or reliability. For example, Xiao *et al.* proposed a new methodology, based on a Dynamic Application Dependency Graph (DADG), to optimize the clustering process [15]. The objectives of this clustering process are to minimize the inter-NoC traffic and to balance the workload between different clusters. First, the DADG is automatically generated from the application using a Low Level Virtual Machine Intermediate Representation (LLVM IR) compiler. Thereafter, a topological sort algorithm is used to map a maximum of one thread cluster to every core within the NoC, such that the two aforementioned objectives are realized. The work is extended in [16] by considering heterogeneous systems and incorporating machine learning techniques. The target systems consists of CPUs, GPUs, and Hardware Accelerators (HWAs). At compile time, Neural Networks (NNs) are used to transform the application into a DADG and extract patterns within the generated graph. Thereafter, a Community Detection (CD) algorithm is used to partition the DADG into communities. At runtime, a dynamic resource manager intelligently maps the resultant communities onto the proper hardware category, i.e., CPUs, GPUs, or HWA. The manager further maps the tasks within each community onto the cores of the target hardware category. The proposed framework proves useful in achieving a good tradeoff between programmability, efficiency, and energy consumption. In general, interested reader in enhancing the performance of NoC-based systems using optimal task-to-core mapping is referred to the surveys presented in [17]–[19].

Dynamic architecture reconfiguration is another widely adopted technique for enhancing the performance of NoC-based systems. In this technique, the underlying architecture is reconfigured according to the executed application at the runtime. For example, Hollis *et al.* changed the topology and the characteristics of a regular mesh to dynamically fit with the traffic requirements of the launched application [20]. The topology adaptation is built on the small world phenomenon [21]. As such, skip-links, or long-range links, are inserted to connect heavily-communicating cores. Experimental results show that the proposed technique successfully reduces the latency and the energy consumption. Xue and Bogdan proposed a general mathematical framework to formulate the dynamic reconfiguration of NoC-based systems [22]. A greedy algorithm is further presented, based on this formulation, to adaptively reconfigure the NoC for lower latency and power consumption. In general, interested reader in enhancing the performance of NoC-based systems using dynamic architecture reconfiguration is referred to the survey presented in [23]. Finally, it is worth mentioning that one problem with the dynamic reconfiguration technique is that it might only work with reconfigurable hardware, such as FPGAs, rather than all types of NoC-based systems.

Using adaptive routing is another widely used technique for enhancing the performance of NoC-based systems. In this technique, packets are adaptively routed through the network in order to avoid creating hotspots within it. For example, Qian *et al.* proposed a hub router and a deadlock-free adaptive routing protocol to enhance the performance of Express Virtual Channel (EVC) NoCs [24]. Presented results show an up to 80% reduction in the latency with a power consumption overhead of only 15%. In general, interested reader in enhancing the performance of NoC-based systems using adaptive routing is referred to the survey presented in [25].

The techniques of traffic shaping and regulation enhance the performance of NoC-based systems by controlling the traffic injection. For example, Lu and Yao presented a dynamic traffic regulation approach to realize the target performance with the minimum buffering requirements [26]. Li and Louri used supervised learning techniques to analyze, predict, and consequently adjust the injected traffic into the network [27]. Traffic shaping and regulation techniques prove useful in enhancing the system performance as well as avoiding network congestion. However, some packets typically suffer from extra delay awaiting the proper network conditions to be injected.

The technique of virtual channel partitioning enhances the performance of NoC-based systems by controlling the assignment of available virtual channels. For example, Jindal *et al.* presented two virtual channel assignment strategies: Augmented Virtual Channel (AugVC) and Output port Directed Virtual Channel (ODVC). Authors also reused trace buffers, which are intended for debugging, to further enhance the performance [28]. Experimental results prove that the presented strategies successfully reduce the latency. Bose and Ghosal combined virtual channel assignment with a novel flow control strategy. A new router design is presented to carry out the assignment of virtual channels and realize the new flow control strategy [29]. Results show a reduction in the area, the energy consumption, the latency, and the packet drop rate. Partitioning the virtual channels between CPUs and GPUs traffic for heterogeneous systems is considered in [30]. Lee *et al.* proposed a feedback-directed virtual channel partitioning mechanism to efficiently share the NoC bandwidth between CPU and GPU cores. Experimental results reflect an increase in the system throughput.

The technique of dynamic resource management enhances the performance of NoC-based systems by optimally allocating the network resources, such as switch crossbar, buffers, and links. For example, Fang *et al.* enhanced the performance of heterogeneous NoC by intelligently allocating buffering resources [31]. Results show a 55.47% reduction in the latency and a 21% reduction in the energy consumption. Qian *et al.* adaptively changed links' direction according to the runtime traffic [32]. A novel router architecture is presented to support this dynamic link customization. Li and Louri used supervised learning techniques to dynamically customize the switch crossbar, the buffer organization, and the routing protocol [27]. The proposed architecture manages to increase the throughput by 28%, decrease the latency and the power consumption by 24% and 19%, respectively.

Network coding is finally used to enhance the performance of NoC-based systems. For example, Xue and Bogdan used network coding techniques to combine multiple packets into a larger one [33]. The work is intended for multicast traffic. The proposed scheme encapsulates cooperation unites, a corridor routing algorithm, and an adaptive flit dropping algorithm to avoid network congestion. Results prove the efficiency of the proposed scheme in reducing the links' utilization and increasing the overall system throughput.

B. NoC INTERFACING TECHNIQUES

Few techniques are presented to properly handle the traffic between interconnected NoC-based systems and efficiently

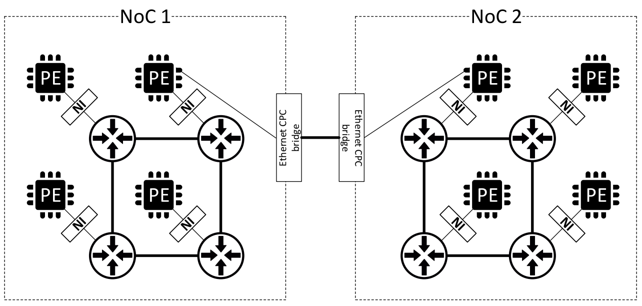


FIGURE 1. Interfacing NoC-based systems using centralized PEs and Ethernet bridges [10].

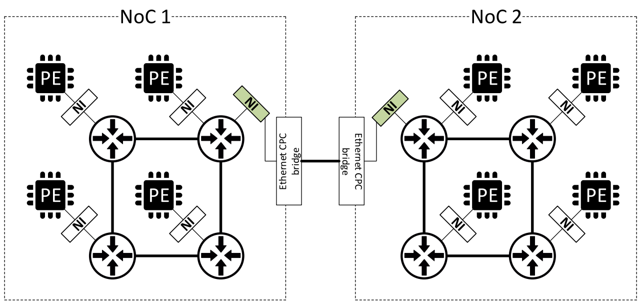


FIGURE 2. Interfacing NoC-based systems using centralized NIs and Ethernet bridges [11].

interface them. In [10], Wasicek *et al.* proposed an NoC-based Ethernet gateway to interface Multi-Processor Systems-on-Chip (MPSoCs). In their proposal, as shown in Figure 1, inter-NoC communication is handled by implementing the gateway logic on a dedicated PE. This PE has an off-system Ethernet bridge attached to it. The gateway PE treats the intra- and inter-NoC traffic differently. It consequently has to convert message formats between the NoC of its own system and that of the interfaced one. Furthermore, due to the use of a centralized gateway, Wasicek's proposal suffers from the drawbacks of the centralized interfacing approach, as mentioned in Subsection III-B. In short, these drawbacks are again creating hotspots in the system, exhausting intra-NoC bandwidth in routing inter-NoC traffic, and necessitating a continuous modification to the gateway PEs or the software running on them.

Nejad *et al.* also proposed interfacing NoC-based systems using off-chip bridges [11]. However, as shown in Figure 2, their proposal relies on an Ethernet CPC connected to a dedicated NI, rather than PE. Accordingly, PEs of a certain system, which wish to communicate with those of other systems, have to address this gateway NI. The intended recipient address should be encoded in the message payload. From the inter-NoC communication perspective, this approach is similar to that of Wasicek. The gateway NI and the bridge are centralized points of communication that could constitute hotspots and prevent the interfacing process from being done optimally. Moreover, PEs of a certain system should be aware of the structure of interfaced systems in order to be able to encode their addresses within messages. This indeed requires modifying these PEs or the software running on them, when a

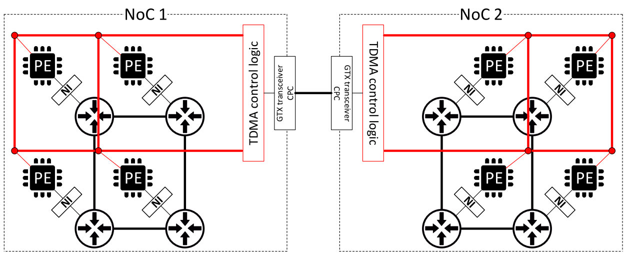


FIGURE 3. Interfacing NoC-based systems using GTX transceivers [12].

new system is connected. Such a modification problem limits the portability of Nejad's proposal and prevents it from being widely used.

Dorai *et al.* proposed interfacing multi-FPGA NoC-based systems using GTX transceivers [12]. As shown in Figure 3, their proposal embeds a GTX transceiver within each NoC-based system. PEs of each system access this GTX transceiver through TDMA control logic. Paths for the inter-NoC traffic, which are drawn in red, are separated from those of the intra-NoC one, which are drawn in black. Although Dorai's proposal is not as centralized as that of Wasicek and Nejad, it still has four drawbacks. First, the TDMA controller constitutes a centralized module in the system and consequently suffers from all associated disadvantages. Second, identifying whether the transmitted packets represent inter-NoC traffic or intra-NoC one is still the responsibility of PEs. This accordingly takes from the computation power of these PEs and reduces their performance. Third, the required TDMA time slices would exhibit a quadratic growth with the increase in NoC dimensions. For large-scale NoC-based systems, this indeed would end up with lots of collisions between these TDMA time slices. Fourth, Dorai *et al.* do not specify how to configure their TDMA time slices. On one hand, these slices could be simply configured in a Round Robin (RR) manner. This could not guarantee the optimal performance under unbalanced inter-NoC traffic. In this article, we evaluate this TDMA-RR approach. Obtained results show that it sometimes gives worse performance than the centralized approach. On the other hand, time slices could be dynamically configured and tuned according to the shape of the inter-NoC traffic using a Traffic Analysis and Queue Prioritization (TAQP) module [34]. This of course would add more complexity to the system and consequently increase its overall latency and power consumption. Alternatively, a static Weighted Scheduling (WS) TDMA could be used. In this TDMA-WS approach, time slices are statically configured according the the bandwidth requirements for each PE. For any application whose inter-NoC communication bandwidth is known, this TDMA-WS emulates the results of using the dynamic TAQP module. Therefore, the TDMA-WS approach is also considered for our evaluation in this article.

In conclusion, previous NoC interfacing approaches could not ensure the optimal handling of inter-NoC communication. In this article, we aim to fill this open research gap by presenting a more efficient interfacing approach, NoC^2. In our approach, NIs are responsible for handling the inter-NoC traffic, not the PEs. To avoid creating a centralized hotspot in the system, the presented approach employs a distributed buffering method, where a dedicated buffer connects between each individual NI and the CPC. Furthermore, the intra-NoC bandwidth is not also consumed in routing the inter-NoC traffic to the gateway PE, or NI. Finally, connecting new NoC-based systems to a one built using our approach does not require modifying any PEs or the software running on them.

VI. NoC^2: AN EFFICIENT NoC INTERFACING APPROACH

NoC^2 is a novel approach for interfacing NoC-based systems and efficiently handling their inter-NoC traffic. The main characteristics of NoC^2 could be summarized as follow.

- It connects NoC-based systems via an interconnection standard. Any standard could be used with our approach. Without loss of generality, an Ethernet standard is employed in this article. Using a standard communication protocol accomplishes three main advantages. First, re-using a widely known and well tested standard facilitates and speeds up the generation of new NoC-based systems, which support productivity and time-to-market. Second, it ensures the portability and compatibility with currently deployed Systems-on-Chip (SoC)-based systems. Third, the evolution of these NoC-based systems would be automatically realized by the evolution of the standard itself.
- NoC^2 requires no modifications to PEs or the software running on them. Once an NoC-based system is designed using our NoC^2 approach, PEs within the system, as well as the software running on them, are completely detached from the inter-NoC traffic management.
- NoC^2 is a general-purpose approach. It puts no restrictions on the interfaced systems, other than using the same communication standard. As will be explained in Subsection VI-C, any new system could dynamically be interfaced to an already running one after completing a handshaking process.
- In order to employ our approach within any NoC-based system, a communication module should be integrated into that system during the design time. This module uses distributed FIFO buffers to efficiently handle the inter-NoC traffic. Therefore, we call it ***I**nter-No**C FIFO*** (ICFIFO). ICFIFO will be discussed in details in Subsection VI-A.

Figure 4 shows the interconnection model of our NoC^2 approach. As shown in the figure, ICFIFO is considered part of the NoC architecture, not just a secondary device attached to a gateway PE. The design of PEs and routers within each tile of the architecture is not affected. Nevertheless, NIs are designed to directly communicate with ICFIFO, instead of routing flits around the network to the gateway PE. NIs are responsible for routing the intra-NoC traffic to the routers within their own tiles and the inter-NoC traffic directly to ICFIFO. Links between NIs and ICFIFO are bidirectional. Moreover, each NI has dedicated buffers inside ICFIFO to allow for simultaneous transmission and reception. As such, the inter-NoC traffic always has a deterministic single hop from the NI to ICFIFO.

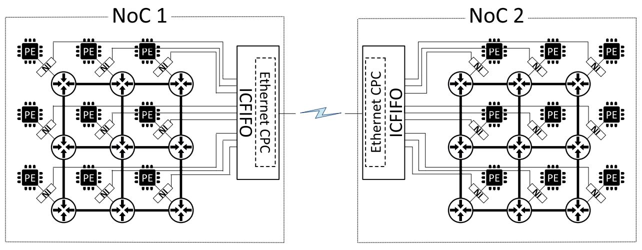


FIGURE 4. ***NoC^2* interconnection model.**

The system-level representation in Figure 4 shows a non-uniformity in the delay of NI/ICFIFO links. A concern that might consequently come to the mind of the reader is whether farther NIs could accomplish the transfer within the single cycle duration. Despite the soundness of this concern and the fact that the actual delays could only be extracted after the floorplanning, the following points verify that this single cycle operation would often be met.

- Previous studies analyzed the delays of global long-range links, based on the small world phenomenon [20], [21]. For most NoC-based systems with typical frequency ranges, these studies confirmed that the delay of these global links would fit within the single clock cycle. They further employed these long-range links to enhance the average latency of the whole system.
- In the actual implementation of the system, the floorplanning and the clock distribution could be adjusted to ensure the single cycle operation and to realize comparable delays over NI/ICFIFO links [35].
- Modern techniques are presented to ensure a single cycle transfer between distant tiles. For example, wireless links are used in [36], optical links are used in [37], and 3D technologies are used in [38]. These techniques could be integrated with our approach to accomplish the required single cycle operation.

In case all of the previous points are not applicable, the pipelining or the Global Asynchronous Local Synchronous (GALS) technique could be used for NI/ICFIFO links. Consequently, the delay of each link would be upscaled according to its length. Even with these extra delays, NoC^2 would still outperform previous interfacing techniques, due to the removal of arbitration, scheduling, and contention delays through routers. To verify the efficiency of our proposed approach for different implementation scenarios, we employ pipelined NI/ICFIFO links in all our simulation results. The number of cycles to traverse each of these links is calculated according to the ratio of its length to that of the inter-router one.

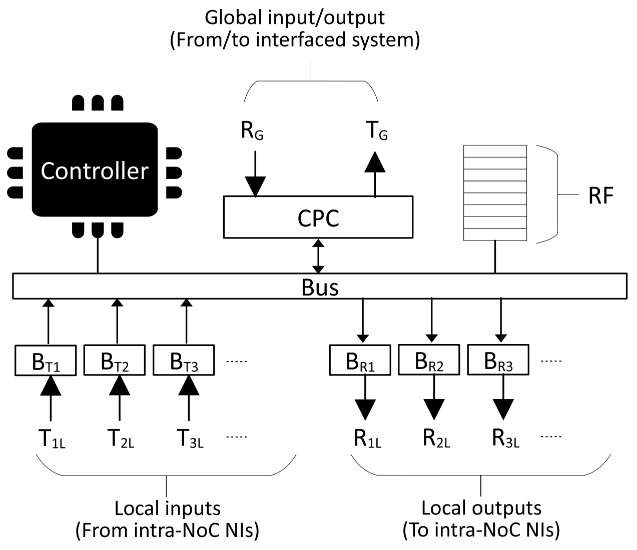


FIGURE 5. Internal structure of ICFIFO.

A. ICFIFO STRUCTURE

ICFIFO is responsible for connecting PEs within its local system to those of another interconnected one. It queues the packets from its local system and initiates the CPC to transmit them. It also receives packets from the other interconnected system through the CPC and writes them into their appropriate NI buffers. Figure 5 shows the inputs, the outputs, and the structure of ICFIFO.

1) INPUTS

The inputs of ICFIFO could be summarized as follow.

- Intra-NoC inputs: ICFIFO has as many intra-NoC local inputs as the number of tiles within its own system. Each NI has a dedicated input for packet transmission, T_{xL}. The traffic comes onto each of these inputs is written into its associated buffer, B_{Tx}.
- Inter-NoC input: ICFIFO has an inter-NoC global input, R_G, which comes from the interfaced system. The traffic comes onto that input is written via the CPC onto the appropriate buffer, B_{Rx}.

2) OUTPUTS

The outputs of ICFIFO could be summarized as follow.

- Intra-NoC outputs: ICFIFO has as many intra-NoC local outputs as the number of tiles within its own system. Each NI has a dedicated output for packet reception, R_{xL}. The traffic transmitted through each of these outputs comes from its associated buffer, B_{Rx}.
- Inter-NoC output: ICFIFO has an inter-NoC global output, T_G, which goes to the interfaced system. The traffic transmitted through this output comes from an NI buffer, B_{Tx}, via the CPC.

3) STRUCTURE

The main components of ICFIFO could be summarized as follow.

- The CPC, which constitutes the interfacing peripheral. An Ethernet CPC is used in this article.
- The controller, which invokes the main functionality of the ICFIFO and controls the CPC.
- The Register File (RF), which holds the status of the ICFIFO and is used as temporary storage for any internal operation.
- The bus, which constitutes the communication medium between different components of ICFIFO.
- The FIFO buffers, which hold packets till being transmitted by the CPC or consumed by NIs. Each NI in the NoC-based system is associated with two FIFO buffers. One is for packets transmission, T_{xL}, and the other is for packets reception, R_{xL}. This is done to allow our ICFIFO to communicate with multiple NIs simultaneously and prevent it from becoming a bottleneck in the system. Compared to the centralized approach, ICFIFO could add its own extra buffers. However, this would increase area and power consumption budgets of the underlying network. Therefore, in this article, we rather relocate some buffers from each NI/router interface to its corresponding NI/ICFIFO interface. As such, the overall number of buffers remains unchanged. For example, if a certain NI/router interface has a buffer size of four packets, we shorten this buffer into two packets only. The saved buffering is then used within ICFIFO for the same NI.

B. NI IMPLEMENTATION

In NoC^2, NI is the module that controls the path for intra- and inter-NoC traffic. This alleviates the burden of controlling these paths from PEs. Our approach employs a typical NI with two slight modifications. First, the controller within the NI is changed to properly adjust the path of the output traffic. Moreover, a special-purpose register is added to every NI in the system to continuously hold the total number of PEs within all interconnected NoC-based systems. Therefore, we call this register the PE Total Number ($PETN$) register. Once the total number of PEs is modified by connecting new systems or disconnecting old ones, the ICFIFO would send a control signal to each NI in its local system with the new number of PEs. In accordance, each NI would update its internal $PETN$ register.

C. NoC^2 OPERATION

1) PROCEDURE FOR INTERFACING NEW SYSTEMS

At startup, the $PETN$ register of each NI is initialized with the total number of PEs within its local system. When another NoC-based system is connected, a handshaking process takes place between the two systems. The ICFIFO of each system transmits a packet to the other one with the total number of its associated PEs. Upon receiving this packet by the other ICFIFO, it sends a control message to its local NIs, containing the new value of the $PETN$ register. NIs update their register in accordance. Finally, each ICFIFO sends an

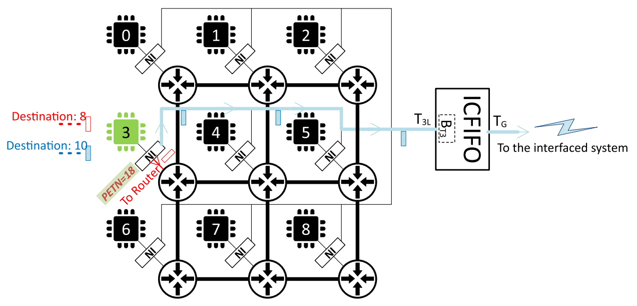


FIGURE 6. An example of egress traffic management.

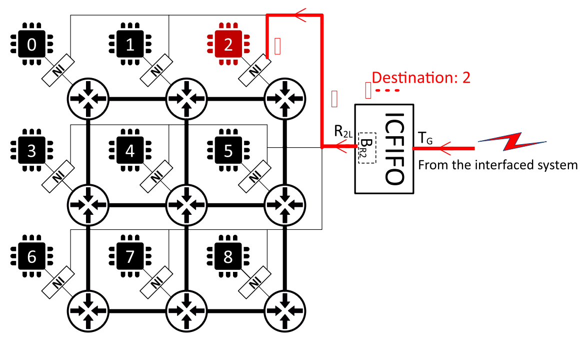


FIGURE 7. An example of ingress traffic management.

acknowledgment signal to the other one in order to terminate the handshaking process and conclude the startup phase. Inter-NoC communication could then take place safely.

2) EGRESS COMMUNICATION

When a PE transmits a flit, the NI examines its header. If the destination ID matches a local PE within its own system, the NI carries out a normal intra-NoC transmission. Otherwise, it queues this inter-NoC traffic into its corresponding ICFIFO buffer, B_{Tx}. Consequently, the ICFIFO dequeues the buffer and triggers the CPC for packet transmission. Figure 6 shows an example of the egress communication. In this example, PE #3 sends a flit, which is colored red, to PE #8 and another flit, which is colored blue, to PE #10. The sender PE is not concerned whether these destination PEs are intra- or inter-NoC ones. Instead, the NI examines each flit header and properly routes it to its corresponding destination. In more details, the destination ID for the red flit matches the ID of a local PE. Therefore, the NI transfers the flit to the corresponding router to issue a normal intra-NoC transmission. On the other hand, the blue flit is destined to a PE within the interfaced system. Thus, the NI writes this flit into its ICFIFO buffer, B_{T_3}. Consequently, the ICFIFO controller triggers a CPC transmission. The CPC encapsulates multiple flits into a single packet, according to the communication protocol standard. This packet is finally sent through the global output, T_G, to the interfaced system.

3) INGRESS COMMUNICATION

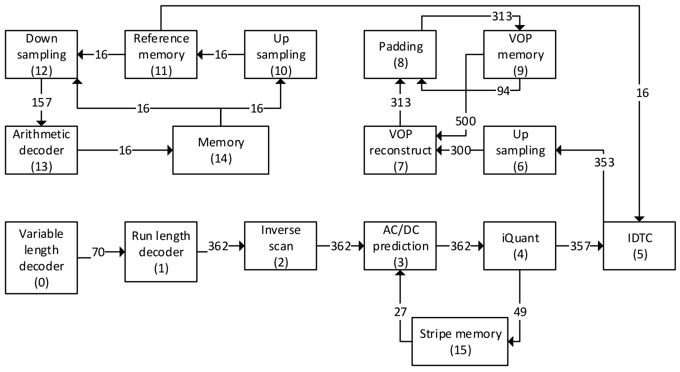
When a packet is received by the CPC from the interfaced system, it is split into flits, according to the communication protocol standard. The ICFIFO controller then examines the header to figure out the destination PE. The traffic is consequently written into the appropriate NI buffer, B_{Rx}. When an NI finds that its R_x buffer becomes non-empty, it forwards the flits from the ICFIFO to its attached PE. Figure 7 shows an example of the ingress communication. In this example, the ICFIFO received a packet from the interfaced system through the global input, R_G. The packet is split into flits and the header is checked by the ICFIFO controller. As the destination is found to be PE #2, the flits are enqueued into its associated ICFIFO buffer, B_{R_2}. NI #2 finally dequeues these flits from the buffer to its corresponding PE.

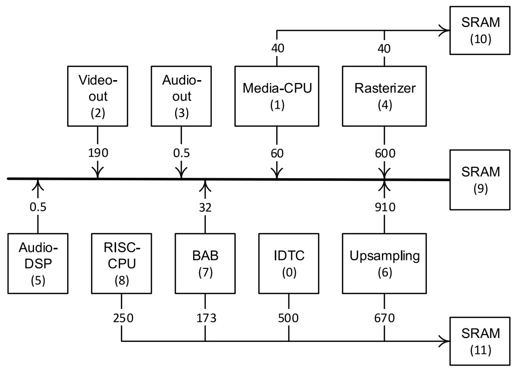
VII. PERFORMANCE EVALUATION: EXPERIMENTAL SETUP

As mentioned in Section IV, our approach is evaluated using synthetic traffic as well as real benchmark applications. The evaluation is done using simulation and hardware implementation. For each considered benchmark, PEs are clustered into two sub-systems. These sub-systems are interfaced to each other using the centralized approach, the two TDMA approaches, and the NoC^2 approach. The performance of these approaches are then compared to one another. As such, in the following subsection, we start our discussion by introducing the considered benchmarks. Thereafter, Subsections VII-B and VII-C describe the employed evaluation environment for simulation and hardware implementation, respectively. For our hardware implementation, we employ an FPGA that runs at 16 MHz, whereas these benchmarks run at 1000 MHz. Therefore, we have to scale down the traffic exchanged between PEs of these benchmarks. Subsection VII-D explains this scaling process in details. Finally, in Subsection VII-E, we discuss the two clustering variants that are considered for evaluating different NoC interfacing approaches.

A. BENCHMARK APPLICATIONS

Many workload modeling methodologies were presented in the literature to generate realistic and scalable benchmarks [39]. Considering the limited size of the employed FPGA, three widely-used NoC benchmark applications are adopted from the literature to evaluate different NoC interfacing approaches in this article. In our experimental work, we will further use an increasing number of PEs and synthetic traffic to evaluate the performance of our approach against the increase in the inter-NoC traffic. The three considered benchmarks are the Video Object Plane Decoder (VOPD) benchmark [40], [41], the Moving Picture Experts Group 4 (MPEG4) decoder benchmark [42]–[44], and the Multi Window Display (MWD) benchmark [43]–[45]. Figure 8 shows the PEs and the bandwidth requirements of each considered benchmark. The VOPD benchmark, as shown in Subfigure 8a, has 16 PEs and an aggregate bandwidth of 3, 574 Mega bits per second (Mbps). The MPEG4 benchmark, as shown in Subfigure 8b, has 12 PEs and an aggregate


(a) VOPD benchmark [40, 41].


(b) MPEG4 benchmark [42–44].

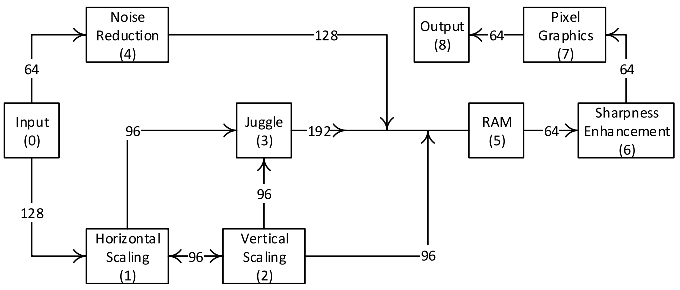

(c) MWD benchmark [43–45].

FIGURE 8. Core graph of the employed benchmark applications (The bandwidth, in Mbps, are written on arrows. The ID of each PE is written inside it beneath its name).

bandwidth of 3, 466 Mbps. The MWD benchmark, as shown in Subfigure 8c, has 9 PEs and an aggregate bandwidth of 1, 184 Mbps.

B. SIMULATION ENVIRONMENT

The extended NoCTweak simulator [46], [47] is used for all our simulation experiments in this article. To the best of our knowledge, this simulator is the only one that allows NoC-based systems to be interfaced to each others with different degrees of inter-connectivity. Accordingly, in our experiments, we first partition every benchmark into two clusters. These clusters are instantiated as standalone systems in the simulator. These systems are then interconnected using the centralized as well as NoC^2 approaches. Unfortunately, the extended NoCTweak does not support the TDMA approach. Accordingly, in this article, this approach is only evaluated through hardware implementation. PEs within each of the two interfaced systems are mapped onto a mesh topology, because it is the only topology that is supported by the simulator. Furthermore, mapping PEs onto this mesh is done using the high-performance bandwidth-oriented mapping heuristic, NMAP [41]. This mapping heuristic is again the one employed by the simulator.

C. HARDWARE IMPLEMENTATION ENVIRONMENT

Figure 9 shows the hardware environment used in our evaluation. For each benchmark, the clusters, which result from the partitioning process, are emulated by traffic generators that run on two hardware modules. These modules are an *Artix-7 35T* FPGA [48] and a Linux-based PC. The two modules are interconnected using an Ethernet link. Due to the size limitation of the employed FPGA, it could maximally emulate six PEs. Therefore, in our partitioning process, we restrict the size of one cluster into six. This resultant 6-PE cluster is executed on the FPGA. The other cluster, which contains the remaining PEs, is executed on the PC. The PC also runs Wireshark [49] for capturing the inter-NoC Ethernet traffic. In order to be consistent with our simulation environment, PEs of each cluster are mapped using the NMAP heuristic onto a mesh topology. Figure 9 shows an example of how the MWD benchmark is executed on the employed hardware environment. The benchmark is partitioned into 6-PE and 3-PE clusters. The 6-PE cluster is executed on the FPGA, whereas the 3-PE one is executed on the PC.

D. BANDWIDTH SCALING

Benchmark applications have communication bandwidth requirements in Mbps, with PEs run at 1000 MHz. In our experiments, PEs are emulated by traffic generators. Based on the employed FPGA, these traffic generators run at 16 MHz. Therefore, the original bandwidth of each benchmark should be scaled down by a factor of 64. This value of 64 comes from ceiling the result of dividing the benchmark execution frequency by that of our FPGA implementation, to the nearest 2^n, i.e., $1000/16 = 62.5$, and the nearest 2^n is 64. Accordingly, the down-scaled bandwidth that is actually generated

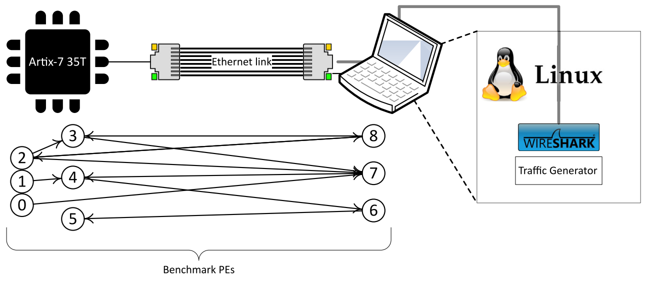


FIGURE 9. **Hardware implementation environment and an example of where the two clusters of the MWD benchmark are executed.**

by our traffic generators is represented as

$$B_{scaled} = \frac{B_{original}}{64} \tag{1}$$

where B_{scaled} is the bandwidth that is practically produced by our traffic generators and $B_{original}$ is the original bandwidth of the benchmarks, as shown in Figure 8. Accordingly, for each benchmark, the traffic generator corresponding to every source PE will inject this scaled bandwidth into the network. For an idealistic scenario, in which this injected traffic experiences no delays in its path, the same bandwidth will be received by the destination PE. This represents the peak throughput, as defined in Section II. This idealistic scenario would indeed never happen practically, due to different types of delays in the network. However, the peak throughput is included in our evaluation results to represent the maximum attainable performance. The closer the throughput of an approach to this peak value, the better its performance is.

In order to be consistent throughout the article, we present all our throughput results in Mbps. However, to accurately describe our implementation, it should be noticed that our traffic generators actually transmit packets, not bit streams. Each packet is composed of 64 bytes. Accordingly, the aforementioned scaled bandwidth, in (1), is segmented into packets. In other words, the scaled bandwidth is divided by 8 to transform it into bytes and the results is further divided by 64 to convert it into packets. Moreover, in order to control and harmonize packets transmission over the execution period, we run each benchmark in a loop that executes 512 iterations per second. Transmitted packets per second should be further divided by 512 to get the number of packets transmitted in every loop iteration. As the repeated division might produce fractions, the resulted number is finally ceiled to the nearest integer. Consequently, the number of packet transmissions that a traffic generator will issue per one loop iteration is calculated as

$$N_{TX_{iter}} = \left\lceil \frac{B_{scaled}}{8 \times 64 \times 512} \right\rceil \tag{2}$$

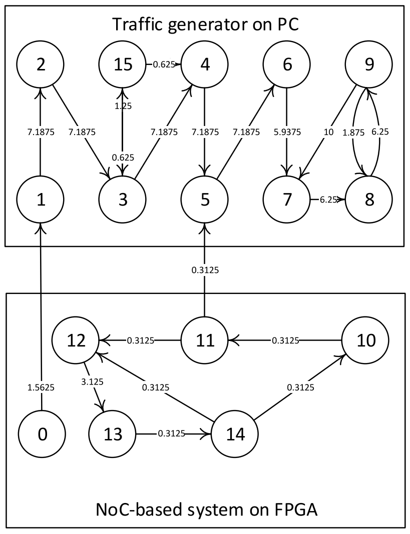
where $N_{TX_{iter}}$ is the number of packets transmitted in a single loop iteration and B_{scaled} is the scaled bandwidth, as expressed by (1). The following example numerically clarifies the scaling process. An original benchmark bandwidth of 64 Mbps is first scaled down to 1 Mbps. This 1 Mbps is further divided into 2048 packets of 64 bytes each. Considering 512 loop iterations per second, 4 packets are actually transmitted every iteration. Based on this scaling, two points could be easily deduced. First, due to the ceiling process, the practically injected traffic might be slightly higher than the scaled bandwidth, as expressed by (1). However, it should be emphasized that this is very minor in a way that would not affect the correctness nor the fairness of our evaluation. Second, the smallest bandwidth that could be produced by any traffic generator is that corresponding to a single packet. This could be calculated as $\frac{1\times512\times64\times8}{1024\times1024} = 0.25$ Mbps of scaled traffic. In turn, this corresponds to an original bandwidth of $0.25 \times 64 = 16$ Mbps. Any bandwidth lower than this 16 Mbps will consequently be treated as a 16 Mbps one. Nevertheless, as shown in Figure 8, we only face this case twice in the MPGE4 benchmark. Again, the correctness of our evaluation is not affected by this insignificant change.

Before concluding this subsection, we need to clarify two more points. First, our simulation is conducted using the original bandwidth, rather than the scaled one. Second, the inter-NoC traffic would specifically be higher than the scaled one, as expressed by (1). This will occur irrespective of the ceiling process and even for our idealistic scenario. It actually happens due to the fact that Ethernet adds additional 16 bytes as control overhead. These additional bytes result in an Ethernet packet length of 80 bytes. Therefore, inter-NoC traffic is specifically scaled up by a factor of 80/64. For example, considering our idealistic scenario of no delays and for a scaled bandwidth of 1 Mbps, the peak observed bandwidth over the Ethernet link would be 1.25 Mbps.

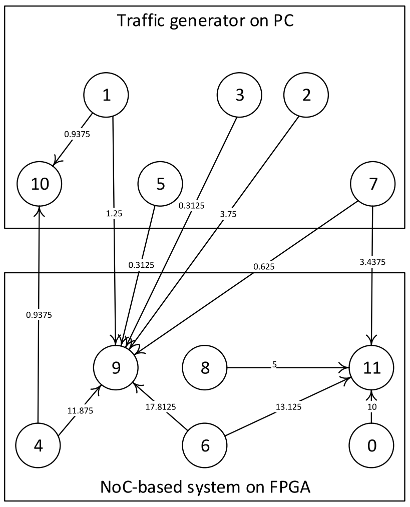
E. CLUSTERING VARIANTS

As mentioned at the beginning of this section, PEs of each benchmark are clustered into two sub-systems, which are then interfaced to each other. In our experiments, we considered two variants for this clustering process. The first variant minimizes the inter-NoC traffic, and hence, maximizes the intra-NoC one. In contrary, the second variant maximizes the inter-NoC traffic, and hence, minimizes the intra-NoC one. Throughout this section, we call these two variants the optimal and the worst clustering, respectively. The rationale behind using these two clustering variants is to ensure that NoC^2 would perform well for any application with low or high inter-NoC communication requirements. On one extreme of a high inter-NoC traffic, we aim to ensure that NoC^2 achieves the best performance with respect to other interfacing approaches. On the other extreme of a high intra-NoC traffic, we aim to ensure that re-allocating some buffers from each NoC to ICFIFO would not degrade the performance of any of the two interfaced systems.

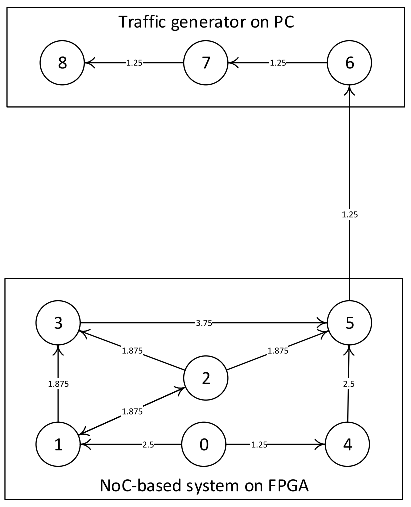
Both clustering variants are realized using the ParMETIS graph partitioning package [50]. ParMETIS automatically generates the optimal clustering by minimizing the total bandwidth, i.e., weight, of the cut edges. For the worst clustering, we negate all the bandwidths and again run the ParMETIS for the minimal total weight. This consequently



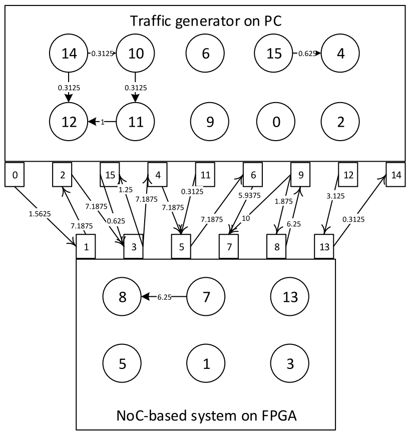
(a) Optimal clustering for the VOPD benchmark.



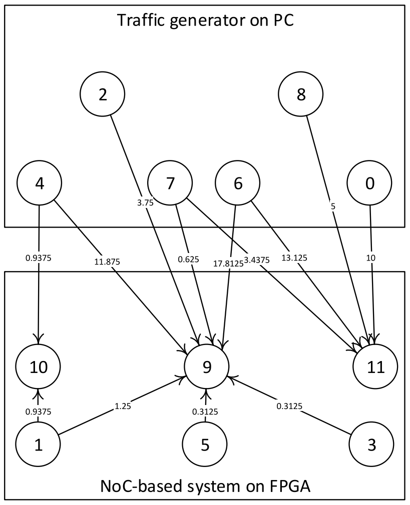
(b) Optimal clustering for the MPEG4 benchmark.



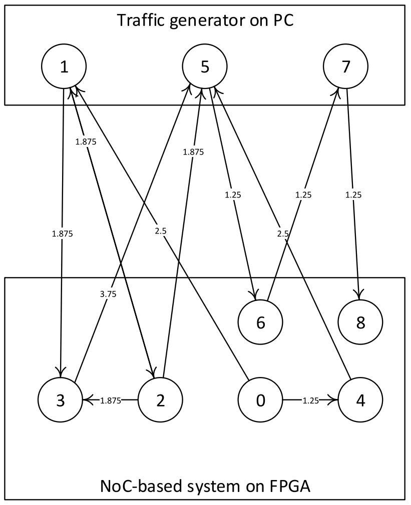
(c) Optimal clustering for the MWD benchmark.



(d) Worst clustering for the VOPD benchmark.



(e) Worst clustering for the MPEG4 benchmark.



(f) Worst clustering for the MWD benchmark.

FIGURE 10. **Optimal and worst clustering of the three considered benchmarks (bandwidth is the scaled one in Mbps).**

generates the worst clustering. For each benchmark, the resultant optimal and worst clustering variants are shown in Figure 10. It is worth mentioning that the figure represents the scaled bandwidths, rather than the original ones. Table 1 summarizes the original and the scaled Ethernet bandwidths of the three considered benchmarks. The figure and the table clearly show that the optimal clustering of the three benchmarks has low inter-NoC traffic and high intra-NoC one. This is apparently reversed for the worst clustering variant.

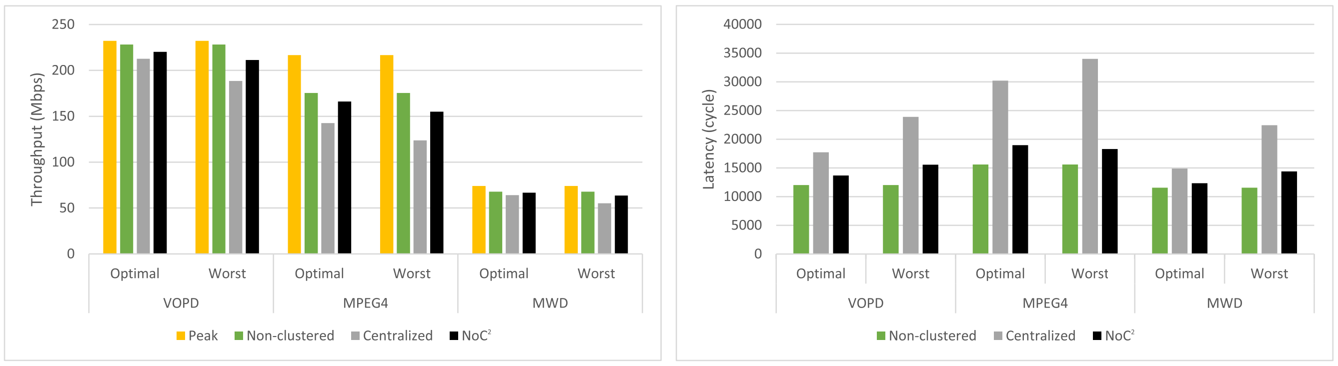
VIII. EXPERIMENTAL RESULTS

In this section, we assess the performance of different NoC interfacing approaches using synthetic traffic as well as the three considered benchmarks. As mentioned in Section IV, our NoC^2 approach is mainly proposed to enhance the throughput and the latency. Therefore, we present many results to extensively evaluate different interfacing approaches with respect to these two metrics. However, due to the importance of power and area to any NoC-based design, we conclude this section by quickly verifying the efficiency of our approach with respect to these two later metrics.

Throughout this section, the optimal as well as worst clustering variants are considered for evaluation. In all our experiments, the simulation and the hardware implementation results are found to be consistent with each others. Therefore, for the sake of being short and concise, we suffice by presenting one of them. First, in Subsection VIII-A, we show and discuss our simulation results for evaluating the overall performance of the whole interconnected systems. This overall performance is then split into its inter-NoC and intra-NoC parts. The two parts are discussed in Subsections VIII-B and VIII-C, respectively. Unlike the

TABLE 1. Original and scaled Ethernet inter-NoC bandwidths, in Mbps, for optimal and worst clustering of the three considered benchmarks.

Parameter	Optimal Clustering			Worst Clustering		
	VOPD	MPEG4	MWD	VOPD	MPEG4	MWD
Transmitted original bandwidth	86	40	64	1455	0	704
Received original bandwidth	0	456	0	1856	3365	320
Total inter-NoC original bandwidth	86	496	64	3311	3365	1024
Transmitted scaled Ethernet bandwidth	1.875	0.9375	1.25	28.1258	0	13.75
Received scaled Ethernet bandwidth	0	9.6875	0	37.1875	66.5625	6.25
Total inter-NoC scaled Ethernet bandwidth	1.875	10.6255	1.25	65.3125	66.5625	20.00



(a) Average router throughput.

(b) Average packet latency.

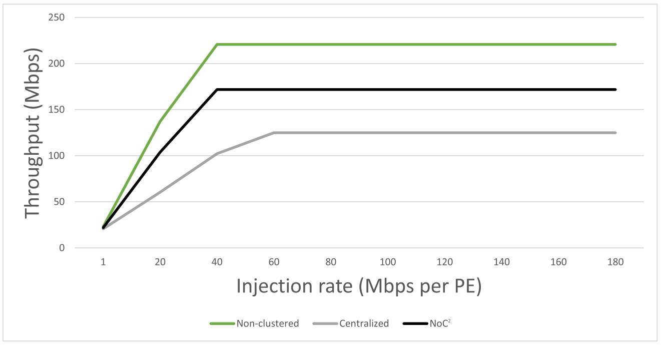
FIGURE 11. Average router throughput and packet latency resulted from different approaches for the two clustering variants of the three considered benchmark applications.

overall performance subsection, the hardware implementation results are shown and discussed in these two subsections. After evaluating the throughput and latency performance completely, we finally verify that our approach does not consume more power than the centralized counterpart and its area overhead could be tolerated. Therefore, in Subsection VIII-D, we present sample results with synthetic traffic and increasing number of PEs to ensure these two points.

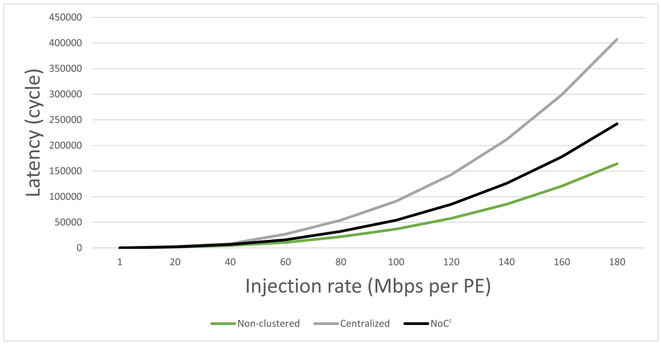
A. OVERALL PERFORMANCE EVALUATION

In this subsection, we evaluate the overall performance of the whole system. In accordance, the two sub-systems resulted from our clustering step is interconnected to each other using the centralized as well as our NoC^2 approaches. Unfortunately, the simulator does not support any of the TDMA approaches. Therefore, their results are not included in this subsection. We further implement a non-clustered architecture in which all PEs within a benchmark constitute only one system, as defined in Section II. This architecture abstracts the NoC customization techniques with minimal inter-NoC traffic, as discussed in Subsection V-A. The non-clustered architecture represents the maximum practical performance that could be attained if the inter-NoC traffic overheads are not existent. Finally, the theoretical peak performance is also included in the comparison.

Subfigure 11a shows the throughput resulted from the four approaches. The presented throughput is an average useful one over all routers within an architecture, i. e., a goodput as defined in Section II. For any benchmark or clustering variant, the peak, non-clustered, centralized, and NoC^2 approaches are represented by orange, green, gray, and black bars, respectively. From these results, we have three observations. First, the results clarify that our NoC^2 approach outperforms the centralized one in terms of average throughput. Second, the figure shows that the throughput of NoC^2 is very close to the maximum practical one of the non-clustered architecture. Except for the MPEG4 benchmark, NoC^2 is also close to the theoretical peak of the hypothetical zero-delay architecture. Numerically, relative to the non-clustered architecture, our NoC^2 approach achieves an average throughput of 96.50%, 92.60%, 94.60%, 88.30%, 98.32%, and 93.82% for VOPD-optimal, VOPD-worst, MPEG4-optimal, MPEG4-worst, MWD-optimal, and MWD-worst, respectively. In contrary, the centralized approach achieves 93.20%, 82.60%, 81.30%, 70.50%, 94.20%, and 81.10% for the six respective cases. These results clearly emphasize the superiority of our NoC^2 approach over the centralized one. Third, the figure shows that the most degraded performance of the centralized approach occurs for the MPEG4 benchmark as well as the worst clustering of the other two benchmarks. According to Table 1, all of these variants have high inter-NoC traffic. Apparently, the gateway PE in the centralized approach creates a hotspot that causes the performance to deteriorate. Moreover, increasing the network load, by routing the inter-NoC traffic to this communicating PE, is another reason that badly affects the throughput of the centralized approach. In contrary, NoC^2 does not create this hotspot and allows direct one-hop connection between NIs



(a) Average router throughput.



(b) Average packet latency.

FIGURE 12. **Average router throughput and packet latency resulted from applying the stress test into a 64-PE system.**

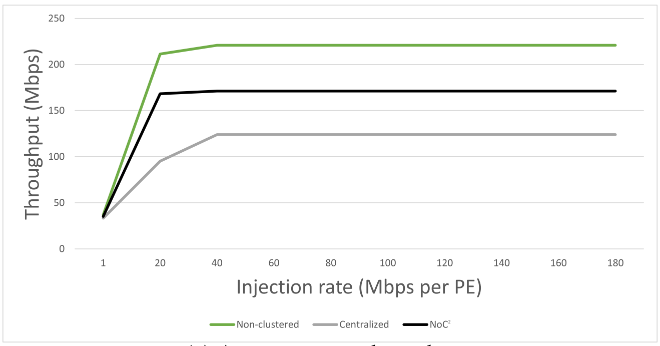
and ICFIFO. Consequently, it manages to attain a good performance for all considered simulation scenarios. This clearly emphasizes the efficiency of our approach in specifically handling heavily-communicating NoC-based systems.

Subfigure 11b shows the average packet latency of different approaches and simulation scenarios. Trivially, the hypothetical zero-delay architecture is not included in the figure. Prior to congestion, it is well known that the throughput and latency are tightly related to one another. The higher the throughput, the lower the latency and vice versa. Therefore, the latency figure shows similar behavior to the throughput one. The figure again emphasizes the superiority of NoC^2 over the centralized approach, especially for heavily-communicating NoC-based systems. It also affirms that NoC^2 manages to achieve an average packet latency close to that of the non-clustered architecture. Numerically, relative to the non-clustered architecture, NoC^2 has an increased average packet latency of 14.00%, 29.60%, 21.60%, 17.20%, 6.72%, and 24.72% for VOPD-optimal, VOPD-worst, MPEG4-optimal, MPEG4-worst, MWD-optimal, and MWD-worst, respectively. In contrary, the centralized approach has an increased average packet latency by 47.50%, 99.00%, 93.50%, 118.00%, 29.00%, and 94.48% for the six respective cases.

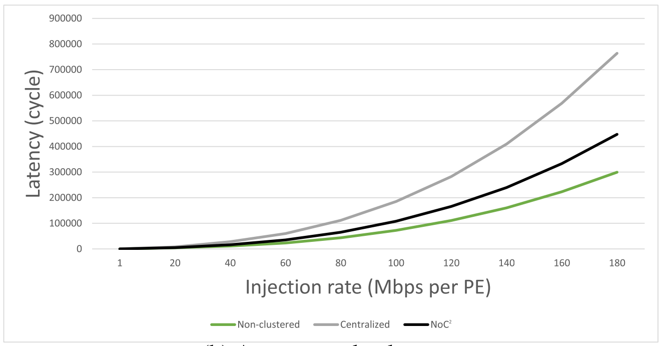
As discussed in Subsection III-A, the number of PEs and the traffic volume for NoC-based systems would rapidly increase in the future. Therefore, we use the stress test technique [51] to evaluate the efficiency of our proposed approach under very high traffic load. In this technique, the NoC is pushed into congestion by continuously increasing the injected traffic. Congestion is identified that the throughput saturates irrespective of the increase in the input traffic. The performance of this stressed network, in terms of throughput and latency, is then measured. Stress test proves useful in validating the NoC performance under extreme traffic conditions. Moreover, in order to verify the performance of our NoC^2 approach for various NoC-based systems, we experiment multiple of these stress tests with increasing number of PEs. We consequently implement different architectures, whose number of PEs increases from 16 and 100. Each PE injects its traffic using a uniform distribution. All results of these different architectures yield the same behavior, with different values. Figures 12 and 13 show two examples of these results for the 64-PE and 100-PE architectures. The 64 PEs are partitioned into 16 2×2 clusters. Similarly, the 100 PEs are partitioned into 25 2×2 clusters. These clusters are interconnected together using our NoC^2 approach as well as the centralized one. A non-clustered architecture is also included in the comparison to represent the ceiling of the practical achievable performance. In terms of average router throughput and average packet latency, obtained results show the superiority of our NoC^2 approach over the centralized one. After going into congestion, NoC^2 could achieve an average throughput that is 47 Mbps higher than the centralized approach. Moreover, for any specific injection rate, our approach could significantly achieve higher throughput and lower latency than the centralized one. Numerically, for post congestion of both the 64-PE and the 100-PE architectures, NoC^2 realizes 78.00% of the throughput of the non-clustered architecture, whereas the centralized approach only realizes 56.00% of it. These results clearly show how powerful is our approach in handling very high traffic load. It reflects the efficiency of our approach in interfacing heavily-communicating NoC-based systems.

B. INTER-NoC PERFORMANCE EVALUATION

In this subsection, we evaluate the inter-NoC performance of different NoC interfacing approaches. Before presenting and discussing our results, three points should be emphasized. First, the non-clustered architecture is not existent throughout this subsection, because its PEs constitute only one system without any inter-NoC traffic. Second, Subsection VIII-A shows that the throughput and latency results are tightly related. We therefore suffice by presenting the throughput results of our hardware implementation. As the simulation and hardware implementation results are consistent, we only include the hardware ones to avoid lengthy repeated discussion. The hardware results allow us to include the two TDMA approaches into our discussion. Third, the throughput results presented in this subsection are the inter-NoC traffic



(a) Average router throughput.



(b) Average packet latency.

FIGURE 13. Average router throughput and packet latency resulted from applying the stress test into a 100-PE system.

passing through the Ethernet link, as measured by Wireshark. This Ethernet throughput of the centralized, TDMA-RR, TDMA-WS, and NoC^2 is presented. Moreover, the theoretical peak inter-NoC throughput of the idealistic zero-delay architecture is also included in the comparison.

Figure 14 shows the Ethernet throughput resulted from the five aforementioned approaches. From this figure, we have five observations. First, the figure clarifies that NoC^2 significantly outperforms the other approaches, specially for the worst clustering variant of the benchmarks. Indeed, the worst clustering has higher inter-NoC traffic and requires an efficient interfacing approach to properly handle that traffic. This agrees with our conclusion at the end of the previous overall performance evaluation subsection that the efficiency of our proposed approach better appears as the inter-NoC traffic increases. Second, for the centralized approach, the gateway PE apparently constitutes a bottleneck in the system that causes the throughput to deteriorate. Third, as the TDMA-RR is not tailored to the traffic, the round robin scheduling dedicates many of the time slots to cores that have no traffic. This wasted time significantly reduces the achieved inter-NoC throughput. Fourth, the TDMA-WS surprisingly achieves an average throughput between 25.00% and 85.00% of the theoretical peak. Although the TDMA-WS is statically tailored to the traffic, random delays within the system shift the transmission of packets for a certain PE from its assigned time slots. This nonalignment not only lowers the throughput, but it also prevents packets from being received in their statically designated time. Destination PEs will therefore stall awaiting for the rest of the packets to arrive, once the source PE is again granted time slots to transmit. Unfortunately, these delays are random and could not be statically predicted. The accumulation of these delays eventually lowers the throughput of the TDMA-WS. Fifth, the figure shows that our approach is the closest one to the maximum attainable throughput of the theoretical peak. Numerically, relative to the theoretical peak throughput, our NoC^2 approach achieves an Ethernet throughput of 92.50%, 92.70%, 94.40%, 85.80%, 98.30%, and 97.80% for VOPD-optimal, VOPD-worst, MPEG4-optimal, MPEG4-worst, MWD-optimal, and MWD-worst, respectively. The centralized approach achieves

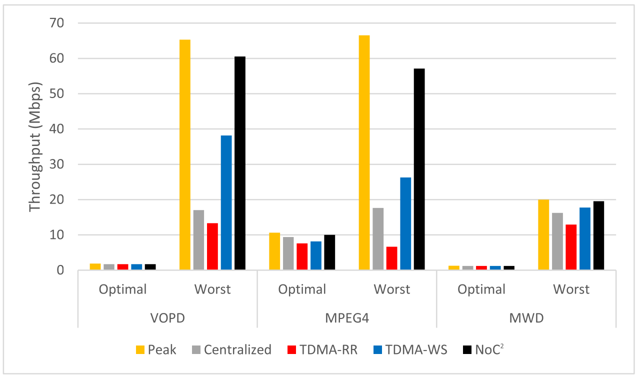


FIGURE 14. Ethernet throughput resulted from different interfacing approaches for the three considered benchmark applications.

90.60%, 26.00%, 88.40%, 26.50%, 98.30%, and 81.10% for the six respective cases. Furthermore, TDMA-RR achieves 92.20%, 20.40%, 71.60%, 10.00%, 98.30%, and 64.60%, whereas TDMA-WS achieves 92.20%, 58.40%, 76.70%, 39.50%, 98.30%, and 88.70% for the same six cases, respectively. In summary, presented results clearly show the efficiency of our approach over other ones in handling the communication between interfaced NoC-based systems.

We again use the stress test to evaluate the performance of various interfacing approaches for future high-traffic NoC-based systems. Therefore, similar to Subsection VIII-A, different architectures are implemented. The injected traffic is uniformly increased and the resultant Ethernet throughput is measured. Sample of the obtained results for the aforementioned five approaches is shown in Figure 15. From these results, we notice that the peak throughput saturates at 80 Mbps. Thereafter, starting from the worst, the centralized, TDMA-RR, TDMA-WS, and NoC^2 approaches saturate at 20 Mbps, 27.5 Mbps, 44.5 Mbps, and 63 Mbps, respectively. In sequence, these values are corresponding to 25.00%, 34.00%, 56.00%, and 79.00% of the peak throughput. Once again, the degraded performance of the centralized approach is due to the competition on the gateway PE and the heavy network load resulted from routing the inter-NoC traffic through each cluster to this gateway PE. The performance deterioration of the TDMA-RR approach returns to the cycles

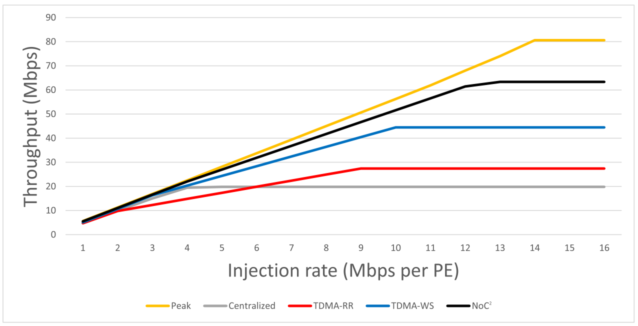


FIGURE 15. Ethernet throughput resulted from applying the stress test into different NoC interfacing approaches.

wasted by a PE pending for a TDMA time slot to be granted. TDMA-WS is affected by the nonalignment between real packets transmission and their statically assigned time slots. In conclusion, from the inter-NoC traffic perspective, results of the stress test clearly proves that the proposed approach is the most efficient one in interfacing heavily-communicating NoC-based systems.

C. INTRA-NoC PERFORMANCE EVALUATION

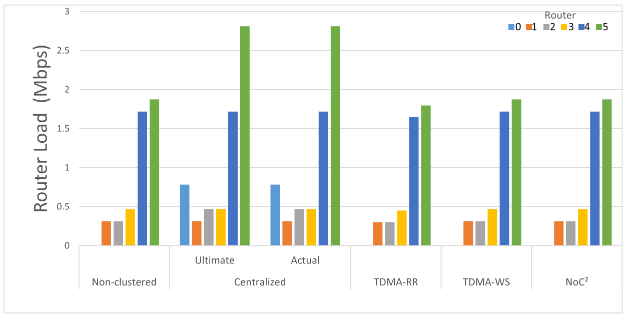
In this subsection, we assess the intra-NoC performance of different interfacing approaches. We herein concerned in ensuring two points. First, we have to verify that the re-distribution of buffers between NIs and ICFIFO in our proposed approach does not affect the intra-NoC performance. Second, for each approach, we evaluate how routing the inter-NoC traffic through a system overloads its intra-NoC routers. Therefore, the router load, as defined in Section II, is used to quantify whether an interfacing approach burdens the routers with extra traffic or not. It is obviously enough to show and discuss the routers load of one of the two interfaced clusters. Throughout this subsection, we present the load of each router within the cluster that is implemented onto the FPFA. In order to measure the routers load on the hardware level, we dedicate a General-Purpose Input-Output (GPIO) pin to every router. This pin toggles for every packet passing through its corresponding router. As we have six routers inside the FPGA, six pins are used. The activity on these six pins is simultaneously monitored by a signal analyzer. Knowing the size of every packet, we then calculate each router load in Mbps.

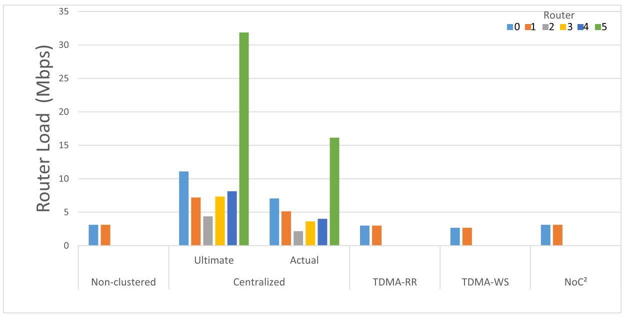
For each benchmark, five interfacing approaches are implemented and considered for evaluation. These approaches are the non-clustered, the centralized, TDMA-RR, TDMA-WS, and NoC^2. On one hand, the non-clustered approach would have the minimum traffic going through its routers, as it has no inter-NoC communication. This provides us with a baseline reference of what the network load should be and how much extra overhead traffic is added by each interfacing approach. On the other hand, for the centralized approach, we present two sets of results. The first, which is named actual, represents the router load that is actually measured on the hardware level. The second, which is named ultimate, is a theoretically-calculated one to indicate the load of each router if we use unlimited buffers and assume a congestion-free network. This ultimate load is included in the comparison to show how worse could the router load reach, as a result of routing the inter-NoC traffic through the network.

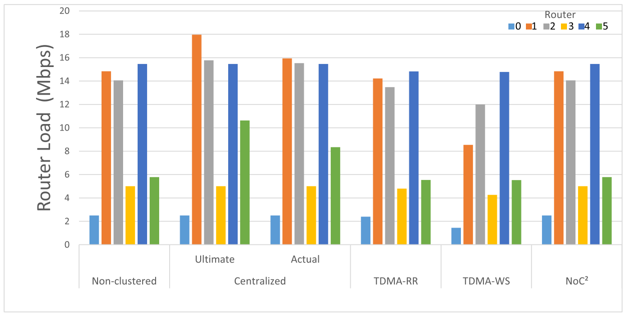
Figure 16 shows the load of every FPGA router, for each benchmark and clustering variant. From this figure, we have three observations. First, the centralized approach has a significantly higher intra-NoC load, compared to others interfacing approach. This increased load is apparently due to routing the inter-NoC traffic through the network to the gateway PE. In turn, this gateway PE, which is represented by the last green bar, has the worst load impact. It receives additional traffic that is equal to the total inter-NoC traffic of all PEs, except itself. This heavy traffic would indeed affect the execution on this PE. Much of its computation power would certainly be dedicated to manage this heavy inter-NoC traffic. Second, the figure clarifies that the actual routers load of the centralized approach is much lower than the ultimate one, specially for the worst clustering variant of each benchmark. On average over the six routers, the actual load is 54.00%, 52.00%, and 59.00% of the ultimate one for VOPD-worst, MPEG4-worst, and MWD-worst, respectively. This observation reflects a significant amount of congestion, which occurs in the centralized approach and prevents its actual load from reaching the ultimate one. The congestion would consequently increase the packet loss rate, the latency, and the power consumption. This observation again emphasizes the importance of our proposed approach for heavily-communicating NoC-based systems. Third, the figure shows that TDMA-RR, TDMA-WS, and our NoC^2 approach do not affect the intra-NoC performance. They have as much load as the non-clustered counterpart. As these three approaches completely separate between the intra- and inter-NoC traffic, they do not increase the intra-NoC load. Most importantly for us, this observation clearly validates that re-distributing buffers in our proposed approach does not degrade the intra-NoC performance of interfaced systems. In a nutshell, the whole experimental results clearly prove the efficiency of NoC^2 in interconnecting NoC-based systems over other interfacing approaches. Without any additional buffer requirements, it significantly enhances the inter-NoC performance, without degrading the intra-NoC one.

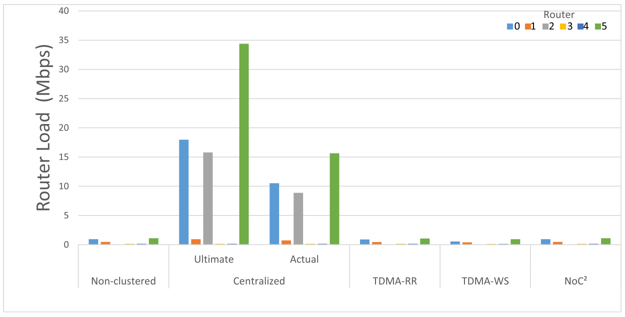
D. POWER CONSUMPTION AND AREA EVALUATION

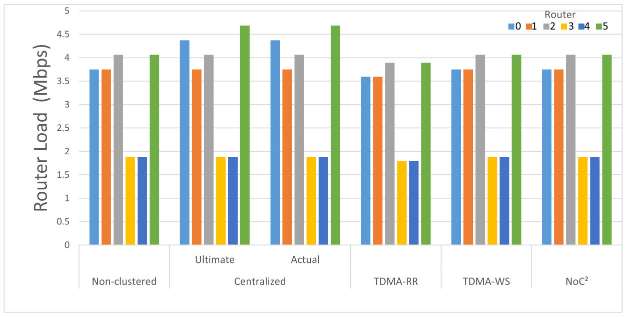
Our approach is mainly proposed to enhance the throughput and latency of future interconnected NoC-based systems. In the NoC domain, any novel approach is not adopted unless its power consumption and area overheads could be tolerated. Our approach requires more area than the currently deployed centralized one due to the extra hardware of ICFIFO, the extra links between NIs and ICFIFO, and the extra port of NIs. Therefore, in this subsection, we quickly evaluate the power and area of our NoC^2 approach versus the current centralized one. Similar to our stress test experiments, we implement


(a) VOPD optimal clustering.


(b) VOPD worst clustering.


(c) MPEG4 optimal clustering.


(d) MPEG4 worst clustering.


(e) MWD optimal clustering.

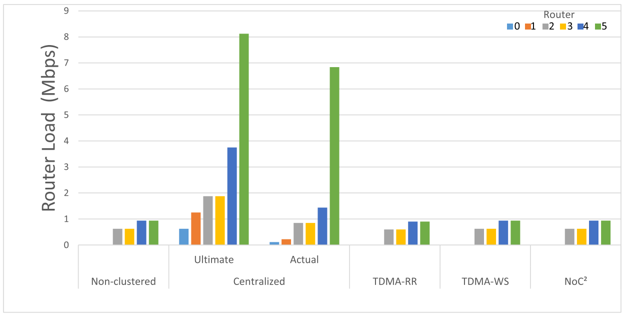
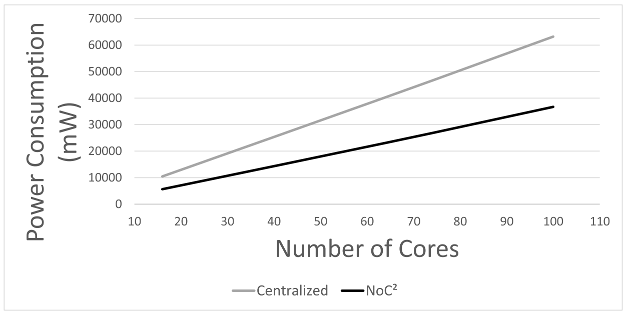

(f) MWD worst clustering.

FIGURE 16. Intra-NoC load for FPGA routers resulted from different interfacing approaches for the three considered benchmark applications.


(a) Avergare power consumption.

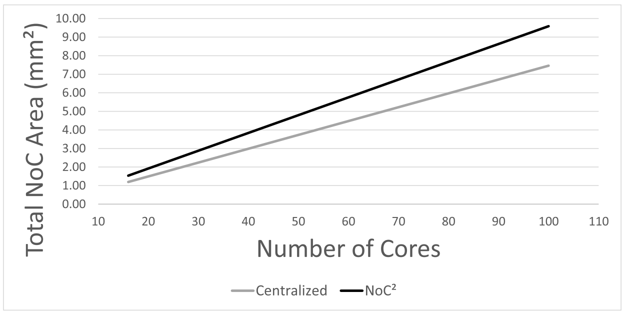

(b) Total NoC area.

FIGURE 17. Power and area comparison between NoC^2 and the centralized approach for different number of cores.

multiple architectures, whose number of PEs increases from 16 to 100. This architectures are partitioned into 2×2 clusters that are interfaced to each others. PEs inject the traffic uniformly with a Flit Injection Rate (FIR) of 0.5. NoCTweak is used to evaluate the total power consumption of the two approaches, whereas ORION system-level area modeling [52] is employed to calculate the total NoC area of them.

Subfigure 17a shows the average power consumption of the two approaches. Throughout the experimented range of PEs, the figure shows that our NoC^2 approach does not burden the power budget of the design. Rather, it significantly reduces the power consumption, compared to the centralized approach. The amount of saved power is further increased as the number of PEs is increased. For example, for 100 PEs, NoC^2 consumes only 58.1% of the power of the centralized approach. This mainly returns to the capability of our approach to avoid congestion, which consumes a noticeable amount of power, as discussed in Subsection VIII-C. These results certainly add another advantage to our approach. Beside enhancing the throughput and the latency, it moreover saves energy.

Subfigure 17b shows the total NoC area of NoC^2 as well as the centralized approach. This total area includes routers, NIs, and links' area. The figure shows that the area overheads of our approach are reasonable. Furthermore, these overheads are not significantly increased as the number of PEs is increased. Considering the significant enhancement realized by our approach for the throughput, latency, and power consumption, these overheads could indeed be tolerated.

IX. CONCLUSION

In this article, we presented a novel approach, NoC^2, to efficiently interface NoC-based systems. Our approach distributes buffers wisely through the interfaced systems. It does not overload the resources within these systems by routing the inter-NoC traffic through them. Once a system is designed according to our approach, its PEs and the software running on them would not be affected by the interfacing process. We evaluated our approach, as well as previous interfacing ones, using synthetic traffic and real benchmark applications. Results show a superior performance of NoC^2 over other approaches, specially for heavily-communicating systems. Compared to the currently deployed centralized approach, NoC^2 significantly increases the throughput and reduces the latency and the power consumption. It does not also suffer from the congestion problem as its centralized counterpart. These enhancements are achieved with a small increase in the NoC area, which could be tolerated. Numerically, compared to the theoretical peak inter-NoC throughput, our proposed approach achieves 79.00% of it. In contrary, the centralized, TDMA-RR, and TDMA-WS approaches only achieve 25.00%, 34.00%, and 56.00% of this peak throughput, respectively. Finally, in the future, we plan to extend NoC^2 to support multiple CPC standards. Also, we would evaluate our approach against dynamic arbitration techniques, such as TDMA-WS with TAQP.

REFERENCES

[1] A. A. Morgan, M. W. El-Kharashi, H. Elmiligi, and F. Gebali, "Unified multi-objective mapping and architecture customisation of networks-on-chip," *IET Comput. Digit. Techn.*, vol. 7, no. 6, pp. 282–293, Nov. 2013.

[2] H. C. Freitas and P. O. A. Navaux, "A high-throughput multi-cluster NoC architecture," in *Proc. 11th IEEE Int. Conf. Comput. Sci. Eng.*, Sao Paulo, Brazil, Jul. 2008, pp. 56–63.

[3] C. Puttmann, J. C. Niemann, M. Porrmann, and U. Ruckert, "GigaNoC–a hierarchical network-on-chip for scalable chip-multiprocessors," in *Proc. 10th Euromicro Conf. Digital Syst. Design Archit., Methods Tools (DSD)*, Lubeck, Germany, Aug. 2007, pp. 495–502.

[4] C.-H. Huang, C.-Y. Chen, and H.-Y. Huang, "Hierarchical and dependency-aware task mapping for NoC-based systems," in *Proc. 11th Int. Workshop Netw. Chip Architectures (NoCArc)*, Fukuoka, Japan, Oct. 2018, pp. 1–6.

[5] R. Manevich, I. Cidon, and A. Kolodny, "Design and dynamic management of hierarchical NoCs," *Microprocessors Microsyst.*, vol. 40, pp. 154–166, Feb. 2016.

[6] A. S. Hassan, A. A. Morgan, and M. W. El-Kharashi, "Introducing NoC^2: Interconnecting NoC-based systems through ethernet," in *Proc. 4th Int. Workshop Design Perform. Netw. Chip (DPNoC)*, Leuven, Belgium, Jul. 2017, pp. 473–478.

[7] J. Nickolls and W. J. Dally, "The GPU computing era," *IEEE Micro*, vol. 30, no. 2, pp. 56–69, Mar. 2010.

[8] B. Bohnenstiehl, A. Stillmaker, J. J. Pimentel, T. Andreas, B. Liu, A. T. Tran, E. Adeagbo, and B. M. Baas, "KiloCore: A 32-nm 1000-processor computational array," *IEEE J. Solid-State Circuits*, vol. 52, no. 4, pp. 891–902, Apr. 2017.

[9] D. Greenfield, A. Banerjee, J.-G. Lee, and S. Moore, "Implications of Rent's rule for NoC design and its fault-tolerance," in *Proc. 1st Int. Symp. Netw.–Chip (NOCS)*, Princeton, NJ, USA, May 2007, pp. 283–294.

[10] A. Wasicek, "Embedding complex embedded systems in large ethernet–based networks," *Network*, vol. 1, no. C2, p. C3, 2011.

[11] A. Beyranvand Nejad, A. Molnos, M. Escudero Martinez, and K. Goossens, "A hardware/software platform for QoS bridging over multi-chip NoC-based systems," *Parallel Comput.*, vol. 39, no. 9, pp. 424–441, Sep. 2013.

[12] A. Dorai, O. Sentieys, and H. Dubois, "Evaluation of NoC on multi-FPGA interconnection using GTX transceiver," in *Proc. 24th IEEE Int. Conf. Electron., Circuits Syst. (ICECS)*, Batumi, Georgia, Dec. 2017, pp. 170–173.

[13] J. Hu, U. Y. Ogras, and R. Marculescu, "System-level buffer allocation for application-specific Networks-on-Chip router design," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 12, pp. 2919–2933, Dec. 2006.

[14] M. Kandemir, Y. Zhang, J. Liu, and T. Yemliha, "Neighborhood-aware data locality optimization for NoC-based multicores," in *Proc. Int. Symp. Code Gener. Optim. (CGO)*, Chamonix, France, Apr. 2011, pp. 191–200.

[15] Y. Xiao, Y. Xue, S. Nazarian, and P. Bogdan, "A load balancing inspired optimization framework for exascale multicore systems: A complex networks approach," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design (ICCAD)*, Irvine, CA, USA, Nov. 2017, pp. 217–224.

[16] Y. Xiao, S. Nazarian, and P. Bogdan, "Self-optimizing and self-programming computing systems: A combined compiler, complex networks, and machine learning approach," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 6, pp. 1416–1427, Jun. 2019.

[17] P. K. Sahu and S. Chattopadhyay, "A survey on application mapping strategies for Network-on-Chip design," *J. Syst. Archit.*, vol. 59, no. 1, pp. 60–76, Jan. 2013.

[18] A. K. Singh, M. Shafique, A. Kumar, and J. Henkel, "Mapping on multi/many-core systems: Survey of current and emerging trends," in *Proc. 50th ACM/EDAC/IEEE Design Automat. Conf. (DAC)*, Austin, TX, USA, May/Jun. 2013, pp. 1–10.

[19] W. Amin, F. Hussain, S. Anjum, S. Khan, N. K. Baloch, Z. Nain, and S. W. Kim, "Performance evaluation of application mapping approaches for Network-on-Chip designs," *IEEE Access*, vol. 8, pp. 63607–63631, 2020.

[20] S. J. Hollis, C. Jackson, P. Bogdan, and R. Marculescu, "Exploiting emergence in on-chip interconnects," *IEEE Trans. Comput.*, vol. 63, no. 3, pp. 570–582, Mar. 2014.

[21] U. Ogras and R. Marculescu, "'It's a small world after all': NoC performance optimization via long-range link insertion," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 7, pp. 693–706, Jul. 2006.

[22] Y. Xue and P. Bogdan, "Improving NoC performance under spatio-temporal variability by runtime reconfiguration: A general mathematical framework," in *Proc. 10th IEEE/ACM Int. Symp. Netw.–Chip (NOCS)*, Nara, Japan, Sep. 2016, pp. 1–8.

[23] H. Leake Kidane and E.-B. Bourennane, "Run-time reconfigurable Network-On-chip: A survey," in *Proc. 15th Int. Multi-Conf. Syst., Signals Devices (SSD)*, Hammamet, Tunisia, Mar. 2018, pp. 846–851.

[24] Z. Qian, P. Bogdan, G. Wei, C.-Y. Tsui, and R. Marculescu, "A traffic-aware adaptive routing algorithm on a highly reconfigurable network-on-chip architecture," in *Proc. 8th IEEE/ACM/IFIP Int. Conf. Hardw./Softw. Codesign Syst. Synth. (CODES+ISSS)*, Tampere, Finland, Oct. 2012, pp. 161–170.

[25] Y. Wu, C. Lu, and Y. Chen, "A survey of routing algorithm for mesh Network-on-Chip," *Frontiers Comput. Sci.*, vol. 10, no. 4, pp. 591–601, Aug. 2016.

[26] Z. Lu and Y. Yao, "Dynamic traffic regulation in NoC-based systems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 2, pp. 556–569, Feb. 2017.

[27] Y. Li and A. Louri, "ALPHA: A learning-enabled high-performance Network-on-Chip router design for heterogeneous manycore architectures," *IEEE Trans. Sustain. Comput.*, early access, Mar. 17, 2020, doi: 10.1109/TSUSC.2020.2981340.

[28] N. Jindal, S. Gupta, D. P. Ravipati, P. R. Panda, and S. R. Sarangi, "Enhancing Network-on-Chip performance by reusing trace buffers," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 4, pp. 922–935, Apr. 2020.

[29] A. Bose and P. Ghosal, "Switching at flit level: A congestion efficient flow control strategy for Network-on-Chip," in *Proc. 28th Euromicro Int. Conf. Parallel, Distrib. Network-Based Process. (PDP)*, Västerås, Sweden, Mar. 2020, pp. 319–322.

[30] J. Lee, S. Li, H. Kim, and S. Yalamanchili, "Adaptive virtual channel partitioning for network-on-chip in heterogeneous architectures," *ACM Trans. Design Autom. Electron. Syst.*, vol. 18, no. 4, p. 48, Oct. 2013.

[31] J. Fang, Z. Chang, and D. Li, "Exploration on routing configuration of HNoC with intelligent on-chip resource management," *IEEE Access*, vol. 8, pp. 12117–12129, 2020.

[32] Z. Qian, S. M. Abbas, and C.-Y. Tsui, "FSNoC: A flit-level speedup scheme for network on-chips using self-reconfigurable bidirectional channels," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 9, pp. 1854–1867, Sep. 2015.

[33] Y. Xue and P. Bogdan, "User cooperation network coding approach for NoC performance improvement," in *Proc. 9th Int. Symp. Netw.–Chip (NOCS)*, Vancouver, BC, Canada, 2015, pp. 1–8.

[34] F. Mashhadi, A. Asaduzzaman, and M. F. Mridha, "A novel resource scheduling approach to improve the reliability of shuffle-exchange networks," in *Proc. IEEE Int. Conf. Imag., Vis. Pattern Recognit. (icIVPR)*, Dhaka, Bangladesh, Feb. 2017, pp. 1–6.

[35] L. Xue, W. Ji, Q. Zuo, and Y. Zhang, "Floorplanning exploration and performance evaluation of a new Network-on-Chip," in *Proc. Design, Autom. Test Eur.*, Grenoble, France, Mar. 2011, pp. 625–630.

[36] K. Duraisamy and P. P. Pande, "Enabling high-performance SMART NoC architectures using on-chip wireless links," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 12, pp. 3495–3508, Dec. 2017.

[37] Y. Ye, J. Xu, B. Huang, X. Wu, W. Zhang, X. Wang, M. Nikdast, Z. Wang, W. Liu, and Z. Wang, "3-D mesh-based optical Network-on-Chip for multiprocessor System-on-Chip," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 4, pp. 584–596, Apr. 2013.

[38] B. K. Joardar, K. Duraisamy, and P. P. Pande, "High performance collective communication-aware 3D Network-on-Chip architectures," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Dresden, Germany, Mar. 2018, pp. 1351–1356.

[39] Y. Xue and P. Bogdan, "Scalable and realistic benchmark synthesis for efficient NoC performance evaluation: A complex network analysis approach," in *Proc. 2016 IEEE Int. Conf. Hardw./Softw. Codesign Syst. Synth. (CODES+ISSS)*, Pittsburgh, PA, USA, Oct. 2–7, 2016, pp. 1–10.

[40] D. Bertozzi, A. Jalabert, S. Murali, R. Tamhankar, S. Stergiou, L. Benini, and G. De Micheli, "NoC synthesis flow for customized domain specific multiprocessor systems-on-chip," *IEEE Trans. Parallel Distrib. Syst.*, vol. 16, no. 2, pp. 113–129, Feb. 2005.

[41] S. Murali and G. De Micheli, "Bandwidth-constrained mapping of cores onto NoC architectures," in *Proc. Design, Autom. Test Eur. Conf. Exhib.*, vol. 2, Paris, France, 2004, pp. 896–901.

[42] E. B. Van Der Tol and E. G. Jaspers, "Mapping of MPEG-4 decoding on a flexible architecture platform," *Proc. SPIE*, vol. 4674, pp. 362–375, Dec. 2001.

[43] S. Murali and G. De Micheli, "SUNMAP: A tool for automatic topology selection and generation for NoCs," in *Proc. 41st Annu. Design Automat. Conf. (DAC)*, San Diego, CA, USA, Jul. 2004, pp. 914–919.

[44] V. Dumitriu and G. N. Khan, "Throughput-oriented NoC topology generation and analysis for high performance SoCs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 10, pp. 1433–1446, Oct. 2009.

[45] E. G. T. Jaspers and P. H. N. de With, "Chip-set for video display of multimedia information," *IEEE Trans. Consum. Electron.*, vol. 45, no. 3, pp. 706–715, Aug. 1999.

[46] A. T. Tran and B. M. Baas, "NoCTweak: A highly parameterizable simulator for early exploration of performance and energy efficiency of networks on-chip," VLSI Comput. Lab, ECE Dept., Univ. California, Davis, Davis, CA, USA, Tech. Rep. ECE-VCL-012-2, Jul. 2012. [Online]. Available: http://vcl.ece.ucdavis.edu/pubs/2012.07.techreport.noctweak/

[47] A. S. Hassan, A. A. Morgan, and M. W. El-Kharashi, "An enhanced network-on-chip simulation for cluster-based routing," in *Proc. 3rd Int. Workshop Design Perform. Netw. Chip (DPNoC)*, Montreal, QC, Canada, Aug. 2016, pp. 410–417.

[48] *Artix-7*. Accessed: Oct. 6, 2020. [Online]. Available: https://www.xilinx.com/products/silicon-devices/fpga/artix-7.html

[49] *Wireshark*. Accessed: Oct. 6, 2020. [Online]. Available: https://www.wireshark.org

[50] G. Karypis, K. Schloegel, and V. Kumar, "ParMETIS: Parallel gragh partitioning and sparse matrix ordering library, version 3.1," Univ. Minnesota, Minneapolis, MN, USA, Tech. Rep. TR 97-060, 1997. [Online]. Available: http://glaros.dtc.umn.edu/gkhome/fetch/sw/parmetis/OLD/ParMetis-3.1.tar.gz

[51] K. Lahiri, A. Raghunathan, and S. Dey, "Evaluation of the traffic-performance characteristics of system-on-chip communication architectures," in *Proc. VLSI Design. 14th Int. Conf. VLSI Design*, Bangalore, India, 2001, pp. 29–35.

[52] A. B. Kahng, B. Lin, and K. Samadi, "Improved on-chip router analytical power and area modeling," in *Proc. 15th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Taipei, Taiwan, Jan. 2010, pp. 241–246.

AHMED A. MORGAN received the B.Sc. degree (Hons.) from the Faculty of Engineering at Shoubra, Benha University, Egypt, in 2000, the Diploma degree in electronic design automation (EDA) and VLSI design from the Information Technology Institute (ITI), Cairo, Egypt, in 2002, the M.Sc. degree from the Faculty of Engineering at Shoubra, Benha University, in 2005, and the Ph.D. degree from the University of Victoria, Victoria, BC, Canada, in 2011. He is currently an Assistant Professor with the Department of Computer Engineering, Cairo University, Egypt. He is currently on leave at the College of Computers and Information Systems, Umm Al-Qura University, Mecca, Saudi Arabia. He has about 25 publications that span journals, conferences, book chapters, and technical reports. His research interests include parallel architectures, multicore systems, digital VLSI design, wireless sensor networks, and networks-on-chip (NoC) modeling, optimization, and performance evaluation.

AHMED S. HASSAN received the B.Sc. degree in systems and biomedical engineering from Cairo University, Egypt, in 2011, and the M.Sc. degree from Ain Shams University, Cairo, in 2018. He is currently working on many-core systems-on-chip (SoC) analysis and design. He is also an Embedded Software Developer, specialized in multicore architecture, wireless connectivity, and automotive Ethernet.

M. WATHEQ EL-KHARASHI received the B.Sc. (Hons.) and M.Sc. degrees in computer engineering from Ain Shams University, Cairo, Egypt, in 1992 and 1996, respectively, and the Ph.D. degree in computer engineering from the University of Victoria, Victoria, BC, Canada, in 2002. He is currently a Professor of computer organization with the Department of Computer and Systems Engineering, Ain Shams University, and an Adjunct Professor with the Department of Electrical and Computer Engineering, University of Victoria. He has published 115 papers in refereed international journals and conferences. He has authored two books and seven book chapters. His general research interests include advanced system architectures, especially networks-on-chip (NoC), systems-on-chip (SoC), and secure hardware. His specific research interests include hardware architectures for networking (network processing units) and security; advanced microprocessor design, simulation, performance evaluation, and testability; and computer architecture and computer networks education.

AYMAN TAWFIK (Member, IEEE) received the B.Sc. (Hons.) and M.Sc. degrees in electrical engineering from Ain Shams University, Cairo, Egypt, in 1983 and 1989, respectively, and the Ph.D. degree in electrical engineering from the University of Victoria, Victoria, Canada, in 1995. He has worked as a Consultant for DND, Canada, and Egetronic, Egypt. He is currently the Head of the Electrical and Computer Engineering Department, College of Engineering and Information Technology, Ajman University, Ajman, United Arab Emirates. He has over 30 years of experience in teaching different academic courses and vast experience in ABET, accreditation, and reaccreditation of electrical engineering programs. He has published more than 60 research papers in renowned journals and conferences. His research interests include digital signal processing, digital image processing, VLSI signal processing, digital communication, the Internet of Things, computer organization, and education technology.