Hindawi
Complexity
Volume 2018, Article ID 1040869, 11 pages
https://doi.org/10.1155/2018/1040869

WILEY | Hindawi

Research Article

Hybrid Network-on-Chip: An Application-Aware Framework for Big Data

Juan Fang, Sitong Liu, Shijian Liu, Yanjin Cheng, and Lu Yu

Faculty of Information Technology, Beijing University of Technology, Beijing 100022, China

Correspondence should be addressed to Juan Fang; fangjuan@bjut.edu.cn

Received 20 April 2018; Accepted 25 June 2018; Published 30 July 2018

Academic Editor: Wei Xiang

Burst growing IoT and cloud computing demand exascale computing systems with high performance and low power consumption to process massive amounts of data. Modern system platforms based on fundamental requirements encounter a performance gap in chasing exponential growth in data speed and amount. To narrow the gap, a heterogamous design gives us a hint. A network-on-chip (NoC) introduces a packet-switched fabric for on-chip communication and becomes the de facto many-core interconnection mechanism; it refers to a vital shared resource for multifarious applications which will notably affect system energy efficiency. Among all the challenges in NoC, unaware application behaviors bring about considerable congestion, which wastes huge amounts of bandwidth and power consumption on the chip. In this paper, we propose a hybrid NoC framework, combining buffered and bufferless NoCs, to make the NoC framework aware of applications' performance demands. An optimized congestion control scheme is also devised to satisfy the requirement in energy efficiency and the fairness of big data applications. We use a trace-driven simulator to model big data applications. Compared with the classical buffered NoC, the proposed hybrid NoC is able to significantly improve the performance of mixed applications by 17% on average and 24% at the most, decrease the power consumption by 38%, and improve the fairness by 13.3%.

1. Introduction

The Internet of Things (IoT) applications, cognitive computing, and cloud computing have seen tremendous growth in recent years [1, 2]. IoT applications use smart sensors to build an Internet of Things [3], which can generate enormous data every hour. In the area of unmanned driving, the system makes a quick decision on the basis of this continual data input, and it demands exascale computing systems to handle these data in a multicore platform [4, 5]. Expected exascale computing systems will have grand capabilities of computation and storage, with many cores on chip and large memory hierarchies. Such a complexity system needs great efforts to guarantee performance efficiency and to minimize power consumption [6, 7]. With the launching of the Intel Xeon Scalable processor [8] for data centres, the network-on-chip (NoC) is generally acknowledged as a "super highway" to increase the bandwidth between on-chip components, reduce latency when accessing spans of memory hierarchy, and improve energy efficiency. However, NoC has suffered severe congestion issues thanks to big data load, causing a great loss in energy efficiency [9, 10]. Heterogeneous NoCs are emerged to tackle the congestion among laminated applications. Since the congestion is subject to the application characteristics, NoC topology, router type, routing algorithm, and congestion control mechanism, we need a comprehensive heterogeneous NoC framework to handle the new challenges. In this paper, we propose a novel hybrid NoC framework that consists of the following:

(1) Hybrid NoCs which combine buffered and bufferless NoCs where buffered NoC refers to its buffered router type and bufferless NoC contains bufferless routers, on account of spectrum characteristics of

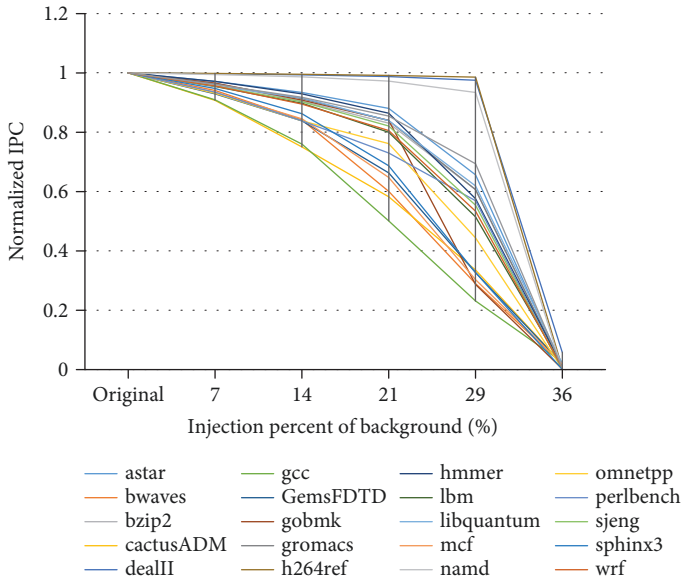


Figure 1: Instructions per cycle (IPC) status with a simulated network pressure.

big data applications, to handle nonuniform resource utilization and large power consumption and supply high efficiency NoCs

(2) An application-aware (APA) mechanism based on the apps' features in real-time, to realize optimal allocation for hybrid NoCs

(3) A congestion optimization algorithm (COA) to monitor the entire system's congestion mode, react to severe congestion, and curb the load

The APA algorithm dynamically analyzes the networks' behavior of all applications and divides the running states of these applications into two parts, that is, latency sensitive (LS) and nonlatency sensitive (NLS). The COA determines the current NoC's resource utilization and whether it is able to facilitate the other NoCs. This approach is effective in dealing with serious congestion problems in isolated NoCs under large and unbalanced loads. Combined with an application-aware policy, it ensures the application performance and system energy efficiency in big data loads and mixed loads.

This paper uses MacSim as the simulation platform, based on a 64-core chip. Compared to the monobuffered scheme, APA enhances the system performance by 4.9% on average (up to 10.4%), reduces latency by an average of 2.4%, and packet deflection by an average of 4.2%. The COA scheme has a performance improvement of 17% on average (up to 24%). Compared to the baseline scheme without congestion control, it reaps a power saving of more than 36%.

2. Problem of Mono-NoC

The traditional buffered NoC handles link contention by using input buffers to store contending flits. It sacrifices a large amount of chip area and power consumption. A buffered NoC does not always work well. When the network load increases, the performance degrades severely. We place the network under pressure by injecting randomly simulated packets into the NoC. The injection percent represents the occupation percent of these random packets. Figure 1 illustrates that as the injection percent increases from 0% to 36%, most apps from SPEC CPU2006 collapse, with the virtual channel (VC) usage of these apps going beyond 50% occupancy in Figure 2. Regardless of the concurrency of the apps, the buffered NoC processes request packets with more cycles when the pressure is high. The buffered network also exhibits a suboptimal performance under high loads. On the other side, a bufferless NoC eliminates the buffers, and flits will be misrouted or dropped if there are link contentions. When the network load is low, misrouting is rare and energy efficiency is good. But one flit may be misrouted for many times at high loads, resulting in performance degradation.

To investigate different apps' performances under big data loads, we explore the VC and link usage of two apps, that is, aster and wrf, which differ significantly in collapse time. Figure 2 illustrates that the link usage of both does increase after reaching 45%, while the VC usage increases evidently. This is because most of the request packets are stored in routers, waiting for a valid output entry. The apps have varying collapse times. For example, namd, when the injection rate is less than 29%, has no significant decline in system performance, while wrf collapses when the injection rate is 29%. This is why the network status affects the performances of the apps as well as the entire system. If this situation occurs on a mono-NoC, the interference among apps would cause some applications not able to deliver their best performances. Applications like wrf are sensitive available network resources.

The proposed hybrid NoC is able to be aware of the application and enhance the energy performance in a fine-grained

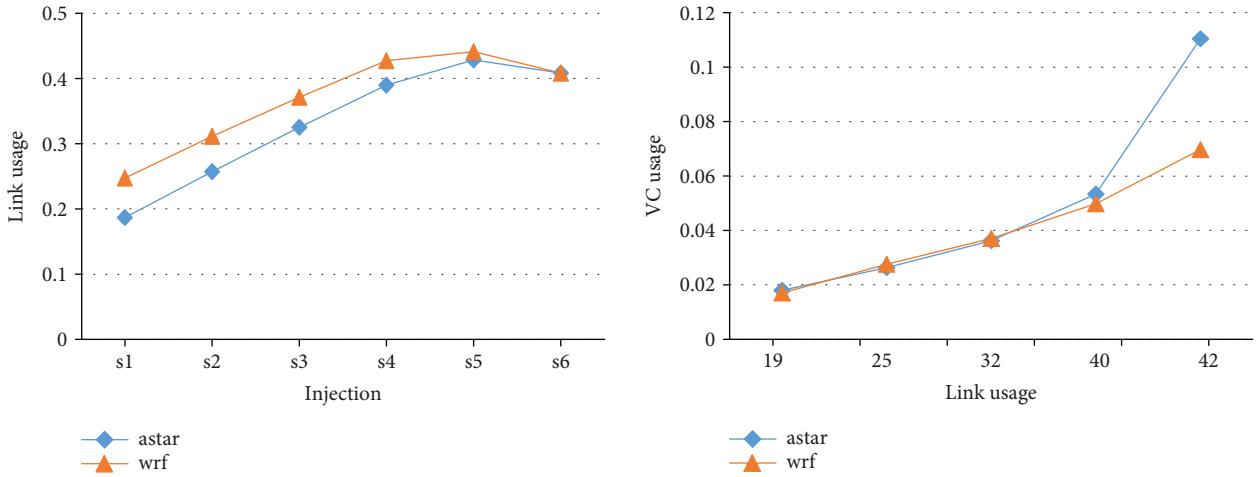


Figure 2: VC usage and link usage with simulated packet injection.

way; the hardware architecture of the hybrid NoC will be explained in Section 4.1. Firstly, we explore that the more suitable packets should be sent to the hybrid NoC and try to classify them in real time. Then we introduce an application-aware mechanism in Section 4.3 and exploit the traffic condition in an effort to balance the load of hybrid NoCs, which will be explained in Section 4.4.

3. Literature Review

With the ever increasing popularity of augmented reality [11, 12] and virtual reality technologies, the design requirements for complex systems with big data are increasing [13]. The one-size-fits-all design approach for the NoC is suboptimal when the network load is heavy, making it difficult to balance between system performance and energy efficiency. Hence, the heterogeneous NoC is emerged to improve energy efficiency. Bokhari et al. [14] present a multimode interconnected architecture to fit requirements for different operational scenarios. Kanoun et al. [15] present a multicore NoC for streaming applications, in which the cores operate as pipeline stages at the classifiers or the feature-extraction level for streaming applications. Ramakrishna et al. [16], Beneventi et al. [17], and Isaac et al. [18] present congestion control mechanisms for the subsequent big load. Mishra et al. [19, 20] present a heterogeneous hierarchical NoC model for diverse types of applications to achieve performance isolation and QoS guarantees. Unfortunately, these state-of-the-art works assume that the applications possess the same network behavior from the start to end, which is not always true, especially for the large array of emerging applications. Besides, they are based on advanced evaluation of applications and thus are not suitable for streaming in the big data era. At last, flow unbalance on different NoCs is not considered. Based upon the above background on the existing work in the literature, this paper proposes a general purpose method for classification, which gives insight into the unbalance situation in real time.

4. Hybrid NoC Design

A hybrid NoC adds a low-power bufferless NoC to a buffered NoC; the architecture of which is illustrated in Figure 3 and consists of two main mechanisms: (1) an application classification mechanism that decides which application is more suitable for either the buffered NoC or bufferless NoC and an injection algorithm for flow control scheme and choosing the NoC layer for flits dynamically, and (2) a dynamical control mechanism that can migrate severe congestion and balance the load. Firstly, we introduce the structure of the hybrid NoC.

4.1. Architecture of Hybrid NoC. The hybrid NoC is based on two NoCs with a 2D mesh topology. The elements are located on the same chip, shown in Figure 3. The communication between two networks is implemented by adding bidirectional links. Two routers located at the same coordinate of the mesh structure share the same processing element (PE), ensuring that the requests from the PE are sent to both NoCs. The network interface (NI) of the hybrid NoC is shown in Figure 4. The traditional single-structure NI uses a miss status handling register (MSHR) to store the status of ongoing requests and injects the requests to the NoC directly. In the hybrid NoC, the application-aware mechanism acquires the information of NC_ratio from the MSHR. Several registers need to be preserved, two of which are used to store the current total network episodes and the episodes' count. The two remaining registers are used to store the current total length of the computing episode, and the episode count of the network episodes occurred. In addition, we also need an extra tag to record the NC_ratio status of the last episode. Until the next cycle, NC_ratio will be overwritten by a new value; otherwise, the router will always refer to the old value of NC_ratio to classify and forward flits.

4.2. Architecture of the Router. Firstly, to enable flit switching between the NoCs, the routing algorithm of a buffered router changes to flit-by-flit routing, instead of worm-hole routing. This implies that each flit should have a control segment to

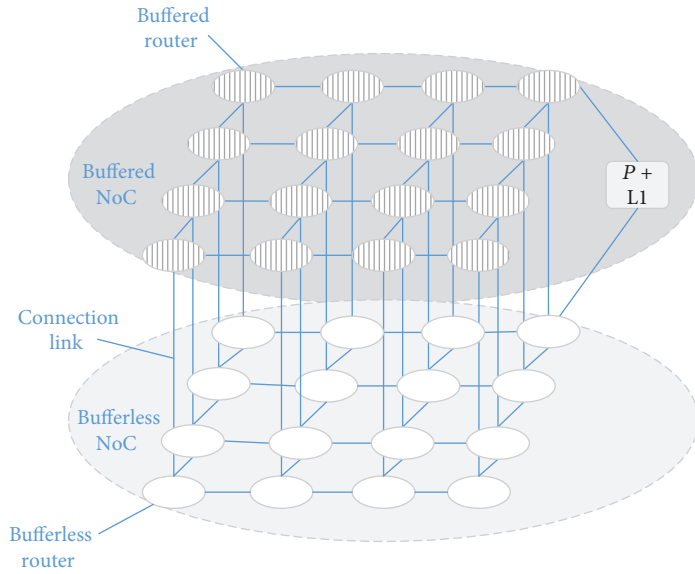


Figure 3: Hybrid NoC architecture.

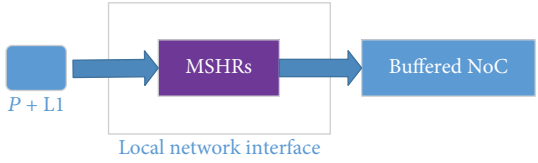


Figure 4: Network interface of the buffered NoC.

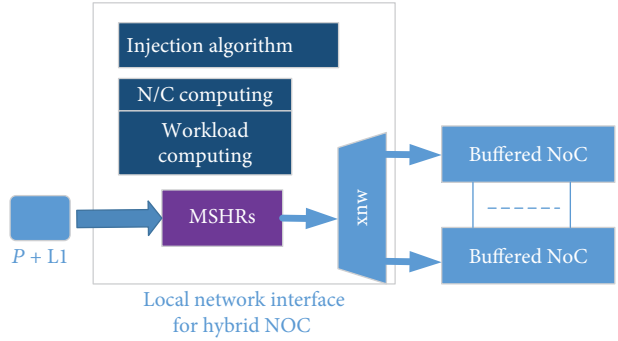


Figure 5: Network interface of the hybrid NoC.

carry the destination information and the sequence number for reassembly. The subnetworks implement switching through multiple sets of routers. One set of routers consists of one buffered router and one bufferless router, so the hybrid NoC can be regarded as 64 sets of routers and other links. Both the buffered and bufferless routers need one more bidirectional link to support flit switching within one set of routers. Secondly, we need to add several registers to the network interface. Four counters are responsible for recording the network episode status, while one register counts the workload status. Figure 4 shows the original NI used in traditional buffered NoC, whereas Figure 5 depicts the modified NI of the hybrid NoC.

4.3. Application Classification Mechanism. When an application is running, the process switches between the network stage and the computing stage. The network stage refers to that at least one of the on-chip network request packet is being processed or waiting to be processed. The request packet is sent to a shared cache after the loss occurred by the private cache and a DRAM. The computing stage is when there are no requests being processed from the network cache or memory. In the network stage, there are multiple outstanding requests from the application of the network in order to implement MLP (memory level parallelism) [3]. If the network stage is longer, the processor will wait for processing the L2 requests and memory requests from the NoC, which means it pauses. At this point, the processor throughput would be low. Moreover, the overall performance could be affected by a lot of request packages, since there could be a number of data packages in the network stage. For the computing stage with a longer period, most of the time the processor is in operation. Due to the very short time for the NoC to obtain data, if those data can be returned quickly, the processor can maximize the advantages of instruction level and memory level parallelism to deliver a better performance. Conversely, if the network performance of these loads decreases, the pause time will be extended to the processor, and then the overall performance of the system is affected. Consequently, it is crucial to classify the applications according their sensitivity to network latency.

Figure 6 illustrates an example of two applications with different network characteristics. Application A contains a long network stage and a relatively short period of the computing stage, while application B has a long period. If these two types of applications run in the same network, using the traditional survival time (age) of data packets to prioritize, the higher age data packets will have a high priority. Imagine a situation in which, as shown in Figure 7, a packet

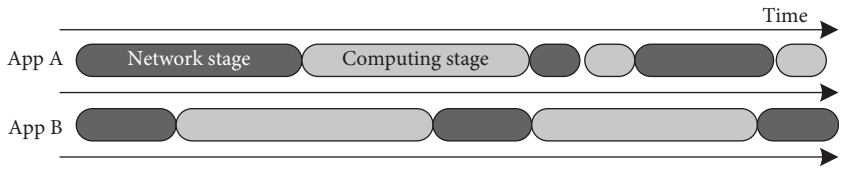
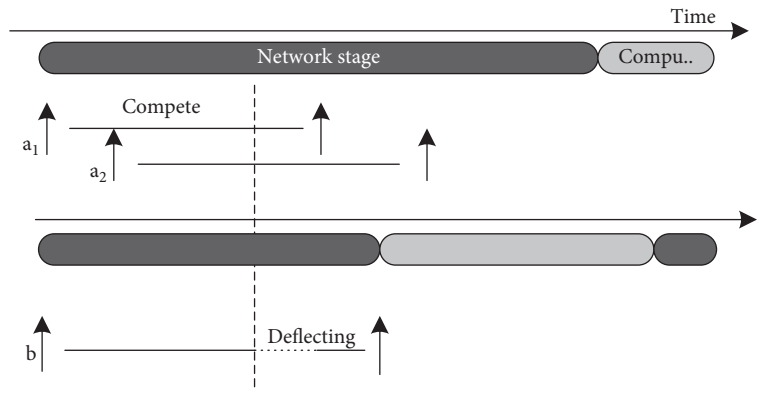
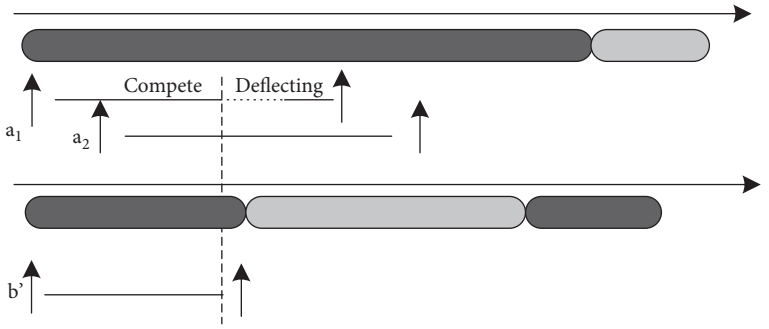


Figure 6: The explanation of the network episode and computing episode.



(a) A way of deflecting



(b) The second way of deflecting

Figure 7: The effect of deflection from different applications.

from application A injects a packet a_1 into the network, and then app B also sends request b to that network. Due to the longer period of the network time of app A, there could be other request packets in the router waiting to be injected. When a_1 and b are in the same port for competition, as a_1's age is older than b, priority will be given to a_1 to use the port. At this point, B will be deflected.

Once a_1 gets the port, it may arrive at the destination faster. However, since app A has a longer network stage, it may have lots of request packages not being returned. App A still cannot get into the computing stage. A single request a_1 has an insignificant effect on its performance, so this application will be classified as nonlatency sensitive. App B has a shorter network stage, which means the latency of a single request package could have larger effect on the overall performance because it could prevent the processor from waiting. Thus this application will be classified as latency sensitive. Therefore, as shown in Figure 7, when in competition with a_1, if b' takes priority over a_1, then the data of application B can be returned as soon as possible, which helps avoid a performance penalty since app B is latency sensitive.

4.4. Application-Aware Classification Mechanism. The network characteristics of the application have a certain locality and phases [21]. Locality refers to local cache access. The application with a large number of shared storage access in the current episode possibly continues to share the storage access in the next procedure. Therefore, we use the network characteristics of the previous episode to predict the network characteristics of the current period. The phase refers to the access feature of application. In other words, at different stages, the characteristics of the application vary, so we take the form of periodic updated parameters. We define NC_ratio (as (1)) to reflect the network feature of different applications in several stages.

$$\text{NC\_ratio} = \frac{\text{AVG}(\text{network\_length})}{\text{AVG}(\text{compute\_length})}, \tag{1}$$

$$\text{AVG}\left(\text{network}_{\text{length}}\right) = \frac{\text{network\_length\_total}}{\text{times}_{\text{episode}}}, \tag{2}$$

$$\text{network\_length\_total} = \sum_{1}^{\text{times}_{\text{episode}}} \left(\text{cycle}_{\text{network}_{\text{end}}}(i) - \text{cycle}_{\text{network\_start}}(i)\right), \quad (3)$$

$$\text{AVG(compute\_length)} = \frac{\text{heartbeat} - \text{network\_length\_total}}{\text{times}_{\text{episode}}}. \quad (4)$$

In (1), NC_ratio refers to the ratio of the average length of the network stages (AVG (network_length), as in (2), it refers to the average length of the computing stages (AVG (compute_length), and as in (3) and (4), it refers the measurement period (heartbeat). The $\text{times}_{\text{episode}}$ refers to the number of measurement cycles of the alternative episode. The start and end of each measurement cycle is not necessarily a complete beginning or ending of the episode, so we set the measurement cycle longer than the average cycle of the alternative episode.

The result in Figure 8 use heartbeats of 10,000 cycles to investigate NC_ratio. If the multicore processor requires a network cycle twice as long as the computing time or less, and if the two cycles of the network application are delayed, the next computing stage will be delayed. We give these requests a higher priority as they are latency sensitive. If multicore requires a network cycle of 10 times longer than the computing cycles, a large amount of data is needed in the computing stage, and overlapping requests in the network are large, plus some of the requested delay, requests can be sent after the deferred compensation. The overall network latency is almost constant, thus classifying this application as nonlatency sensitive. Here, we take NC_ratio = 2 as the cut-off point for the two types of requests. The request can be divided into two classes, that is, latency-sensitive (NC_ratio ≤ 2) and nonlatency-sensitive applications (NC_ratio > 2). The average NC_ratio values of DealII, namd, AStar, h264ref, HMMer, xalancmbk, gobmk, sjent, and bzip2 show that these applications are latency sensitive.

4.5. Application-Aware Mechanism in the Hybrid NoC. To further verify the energy efficiency of latency-sensitive and nonlatency-sensitive requests in both NoCs, we tested the SPEC CPU2006 benchmarks based upon 64 threads. Energy efficiency refers to the throughput per unit of power. The higher the energy efficiency is, the longer the power can last. To identify the differences in energy efficiency between the buffered and bufferless NoCs, we test the energy efficiency of B and BL with 21 benchmarks. Figure 9 shows that the apps running on BL like dealII, namd, h264ref, and hmmer are more energy efficient than running on B. Besides the lower power consumption on BL, the performance of these apps on BL is not significantly affected by nonbuffer routing. Therefore, the energy efficiency of aforementioned apps is significantly higher in the bufferless NoC than in its buffered counterpart. We affirm this type of loads has a lower requirement in bandwidth, so it will be in a good play in the bufferless NoC since it has lower bandwidth as the property.

The result is consistent with the application type divided by NC_ratio. This is because NC_ratio's results reflect the

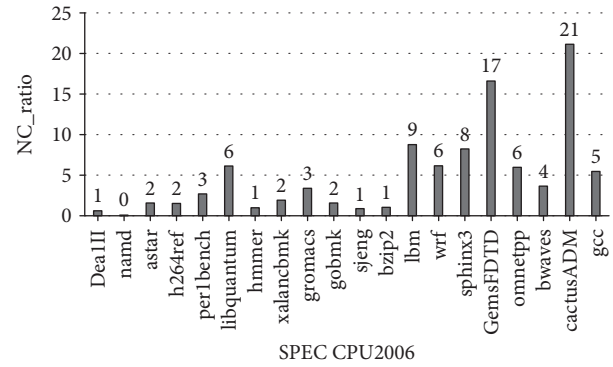


Figure 8: Average NC_ratio of SPEC CPU2006.

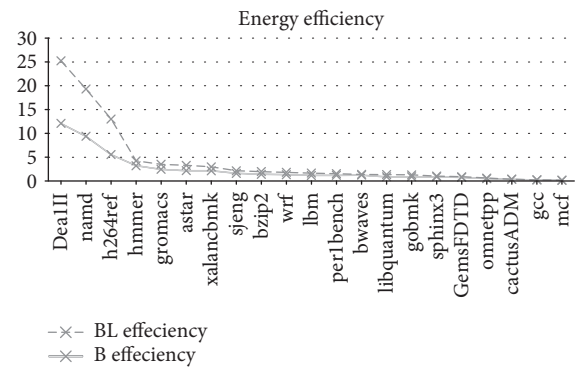


Figure 9: Energy efficiency of applications in the buffered (B) and bufferless (BL) networks.

memory and network resource characteristics of the request. For requests with a lower requirement in network resources, the buffered NoC increases power consumption instead of improving performance. For requests with a higher requirement in network resources, especially large bandwidth requirements, adding more buffers would achieve the same effect as increasing the bandwidth of the network. Which improves the performance greatly, although the power consumption also increases, this can be seen in Figure 9 where the curves are very close. Therefore, in the hybrid NoC, NC_ratio can be used as an "application-network" classification parameter.

During an application's lifespan, it alternates between two episodes, namely, the network episode and the computing episode. The network episode is a period when the application has at least one packet in the network, whereas the computing episode is a period, in which there are no outstanding requests in the MSHR. Figure 8 illustrates that NC_ratio ranges from 0 to 21 except *mcf*, which is an application that has one cycle of the computing episode and 245 cycles of the network episode on average. The apps that have NC_ratio less than one means that the network episode cycles are shorter than the computing episode cycles during the lifespan on average. When the network episodes are twice as long as the computing episodes, it will gain a better performance in BL than B, since the flits in BL will be processed due to nonbuffer and

```
Initialization: NC_ratio = 0, per NI in router; BL = 1, B = 0;
if NC_ratio ≤ 0,
        flit inject layer = cycles % 2;
else/* the first NC_ratio is calculated after heartbeat cycles (1000), and then flushed every
       heartbeat. Heartbeat can be set dynamically */
        if NC_ratio ≤ α, flit inject layer = BL;
        else
           flit inject layer = B;
```

Pseudocode 1

Firstly, an application classification mechanism is used to inject flits to subnetworks. The flits injected into the bufferless NoC are termed flit_{bl}, while the flits injected into the buffered NoC are dubbed flit_{b}.

Compute the local contention state of two subnets. The average starvation ratios σ, δ within t cycles of the bufferless and buffered NoCs can be achieved by and, respectively, where i is the router id.

$$\sigma = \frac{1}{t}\sum_{\Delta t=0}^{t} \text{starved}(i) \in [0,1], \quad (\delta = \sigma),$$

$$\text{starved}(i) = \frac{\text{flits}_{\text{starved}}}{\text{flits}_{\text{throughput}}}.$$

Rate the contention state in step 2, reflected in Table 1.

Every t cycles, retest:

If state_{bl} = Heavy, state_{b}! = Heavy, $\text{flit}_{\text{bl}}(i) \subset [t, t+\Delta t] \rightarrow$ Buffered to (b), otherwise do step 5.

In each cycle, check the buffered router $i+1$ corresponding to router i with state_{bl} = Heavy. If $\text{VC}(i+1)$! = full, flit transmits to the buffered subnet. Otherwise, flit maintains in the current subnet, waiting for another cycle and redo (b) until Δt ends.

Parallelly execute with (a). If state_{b} = Heavy, State_{bl} = Light, check state_{b} the bufferless router corresponding to router i, if it is free for receiving the flit from the buffered router $\text{Bone}(i)$ = empty, $\text{flit}_{\text{b}}(i) \subset [t, t+\Delta t] \rightarrow$ Bufferless, to (d). Otherwise, do step 6.

Within each cycle, check the bufferless router corresponding to router $j+1$ with state_{b} = Heavy, if it is free for receiving the flit from buffered router. $\text{Bone}(j)$ = empty. If yes, inject the flit to port 5, else go to next cycle and redo (d) until Δt ends.

flit_{bl} that were injected into bufferless subnet flit_{bl}, continually transmit in this net, $\text{flit}_{\text{bl}}(i) \subset [t, t+\Delta t] \rightarrow$ Bufferless.

flit_{b} that were injected into buffered subnet, continually transmit in this net, $\text{flit}_{\text{b}}(i) \subset [t, t+\Delta t] \rightarrow$ Buffered.

Algorithm 1: The COA.

misrouting in BL. Here, we set two inequalities for division: NC_ratio $> \alpha$ indicates network intensive and NC_ratio $\leq \alpha$ indicates computing intensive.

Parameter α can be configured in advance. In this paper, we set α to two cycles, so that computing-intensive applications can be distinguished from network-intensive applications. A computing-intensive application means the app will perform better, if available computing resources increase. By contrast, a network-intensive application means the app requires a large number of network operations on-chip, and the performance is greatly affected by the contenting traffic. The bufferless NoC provides a limited but loose network situation for computing-intensive applications (Pseudocode 1).

4.6. Congestion Optimization Algorithm. Network congestion is a phenomenon that multiple packets contend for some ports of the router. In a hybrid NoC, packets are divided into two subnetworks according to the apps' network property; it causes disproportion when there is a capacity gap between two subnetworks. In this part, we present a congestion optimization mechanism to optimize network congestion based on our proposed hybrid NoC, which monitors the resource utilization of the subnetwork and reschedules the network load between the two subnetworks. In this strategy, we need to measure the load of the local network properly, and packets would be transferred to a subnetwork, only if it has extra network resources.

The bufferless NoC is usually of low load. As a result, only the load is relatively low; the bufferless NoC is capable of helping the buffered subnetwork to address capacity issues. When multiple types of flits are in the bufferless subnetwork, the packet priority strategy is an optimized choice. Compared with the bufferless NoC, the buffered subnetwork has a good tolerance of heavy loads. Therefore, when there is serious starvation in the bufferless NoC, many packets wait to be sent by the socket ports in use. Traditional solutions throttle flow, passively waiting for free ports to improve throughput. In this paper, we take the advantages of multiple networks and propose to use the buffered NoC to relieve the starvation pressure of the bufferless NoC. Meanwhile, the bufferless NoC will help the buffered NoC to relieve the power pressure. The cost of transition is one cycle.

4.6.1. Some Note Points. The buffered routing algorithm is improved. Most buffered NoCs adopt worm-hole routing, which splits a packet into several flits, one of which is

Table 1: Partition of congestion degree in hybrid NoCs.

	Bufferless NoC	Buffered NoC
Light	$\sigma < \alpha_{bl}$	$\delta < \alpha_b$
Moderate	$\alpha_{bl} \le \sigma \le \beta_{bl}$	$\alpha_b \le \delta \le \beta_b$
Heavy	$\sigma > \beta_{bl}$, starved	$\delta > \beta_b$, starved

Table 2: System parameters.

	Buffered NoC/bufferless NoC
Topology	2D mesh, 8×8 size
Routing algorithm	X-Y routing, flit-by-flit
Routing latency	2 cycles
Core	Out-of-order, 16 MSHR, 128 instruction windows size
	L1 I-cache and D-cache: 32 KB, 64 B line-size, 2-way, LRU, 2-cycle latency
L1 cache	Private
L2 cache	Per-block interleaving shared, distributed, 64 B line-size, perfect

regarded as the head flit and carries the destination node information. At the arbitration stage, once the head flit receives a pass permit, the flit from same packet will be transmitted in sequence. The bufferless NoC adopts flit-by-flit routing with each flit carrying the destination information. In order to transmit the flits between two nets, both of them adopt flit-by-flit routing. For the buffered NoC, flit-by-flit routing may cause an older flit to wait much longer to receive a pass permit in a channel. This is because in the buffered channel, each flit has a timestamp for arbitration, and general arbitrators choose the winner based on the top flit in each channel. The older and rear flits passively sleep and seriously affect the packet latency. Therefore, the arbitration program in the buffered NoC should consider the ages of all the flits in one channel.

The out-degree is equal to the in-degree in the bufferless NoC to ensure hot-potato routing. Thus, to enable the recept of flits from the buffered NoC, we add an exclusive buffer to temporarily store the packets from the buffered NoC and distribute them as soon as there is a free output port.

5. Methodology

We evaluate the proposed hybrid NoC using MacSim, which is a trace-driven, cycle-level, heterogeneous architecture simulator. MacSim models a detailed pipeline, a memory system that includes caches, NoC, and memory controllers. It also enables multithreading processing. We model an 8×8 processor system based on the mesh topology, with 8×8 buffered routers and 8×8 bufferless routers. Table 2 summarizes the key parameters of our simulated system. The hybrid parameters are as follows: hybrid uses 6 VCs with 5-flit deep buffers for the buffered NoC. We run 1 million cycles for each experiment. Table 3 shows the proportion

Table 3: Mixed nonlatency-sensitive and latency-sensitive loads.

	64 threads	128 threads
W1	50 : 14	100 : 28
W2	40 : 24	80 : 48
W3	30 : 34	60 : 68
W4	20 : 44	40 : 88
W5	10 : 54	20 : 108

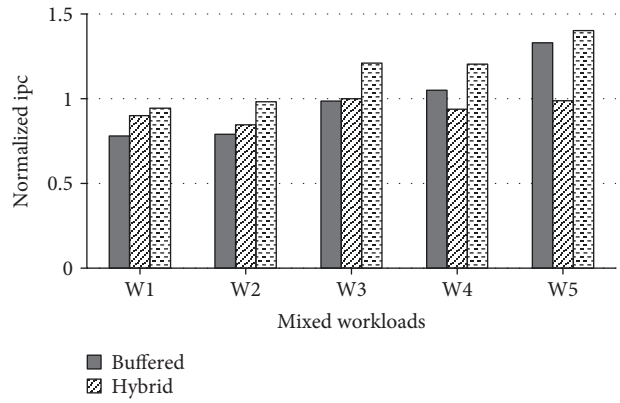


Figure 10: Normalized IPC of hybrid load with 64 threads.

changes with mixed loads, which is used to verify whether the COA is useful for unbalanced loads.

6. Evaluation and Discussions

In this section, we evaluate the performance, power, scalability, and fairness of the buffered and hybrid NoCs. We mix 21 benchmarks and run three copies of each benchmark. The system tests 64 threads with one thread per core.

6.1. Analysis on the Performance and Power Consumption. Figure 10 shows the results of the performance and power on the buffered NoC, hybrid NoC, and hybrid NoC with the COA algorithm. The overall performance of the hybrid COA system is improved by 17% compared to that of the buffered NoC, and the highest is 24%. The performance of the hybrid NoC is improved by 7% compared to the baseline buffered NoC, while this improvement is limited. For example, the performance decreases for W4 and W5 loads. The reason for this is that the mixed composition of latency-sensitive and nonlatency-sensitive packets causes a severe imbalance, which is evident in the subnetworks. The single network, which is a symmetrical structure, automatically adapts to different nodes with varying loads and eventually forms centre and diagonal position traffic characteristics. A heterogeneous network in isolation is easy to form an unbalanced network load, leading to performance degradation. Therefore, the provision of a subnet traffic and congestion optimization mechanism is imperative.

Figure 11 depicts the deflection rates of the bufferless network before and after employing the COA policy. With the COA policy, deflection is significantly improved by an average of 10%. This is because the COA strategy is based on

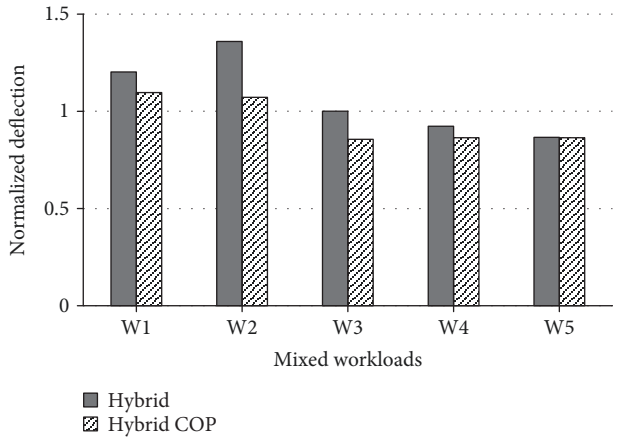


Figure 11: Normalized deflection rate of the hybrid loads with 64 threads.

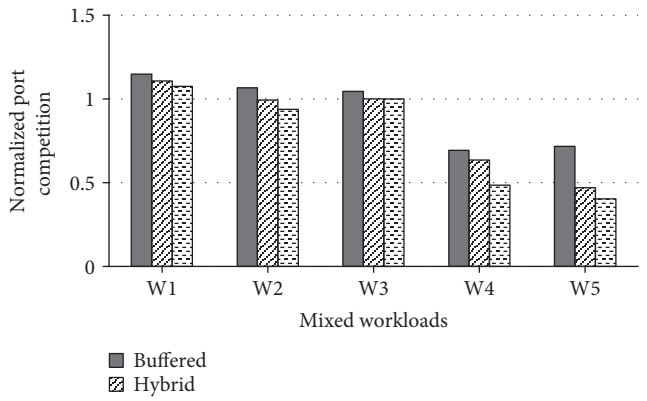


Figure 12: Normalized port competition rate of hybrid loads in 64 threads.

the optimization of starvation. The bufferless subnetwork does not need to be queued for processing requests when the network congested, so that extra request packets have opportunities to be sent to the buffered NoC for the network to reach. This helps save the bandwidth of the inherent bufferless NoC. Therefore, the deflection of the bufferless network has also been improved correspondingly.

Figure 12 shows the port competition of a single buffered NoC and the buffered subnetwork in the hybrid NoC. The hybrid NoC reduces the competition on the single buffered NoC, and the port competition is reduced by an average of 12%. Hybrid COA is more prominent, reduced by an average of 21%. Meanwhile, COA is more remarkable when the load is highly unbalanced in W4 and W5.

Figure 13 compares the hybrid and buffered NoCs in the sense of power consumption. Compared to the network with a buffer, power saving of the hybrid NoC is more than 36%. Hybrid NoC combines a buffered network with a bufferless network so as to take advantages of both networks to improve the system performance. Meanwhile, the power consumption is lower than the simpler buffered NoC.

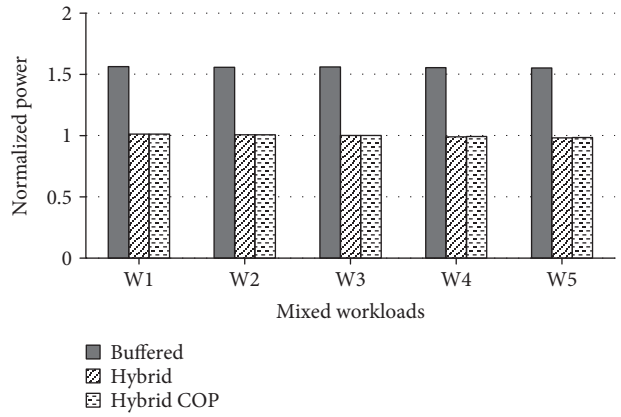


Figure 13: Normalized energy of the hybrid loads with 64 threads.

Figure 14: Normalized IPC of hybrid loads with 128 threads.

6.2. Analysis on Scalability. This section uses the 128 threads, two threads per core for scalability testing. Figure 14 shows the experimental results of IPC in system throughput. In the W1 and W2 mixed mode, the hybrid scheme without COA can improve by 6% on average. This is because the hybrid NoC liberates the latency-sensitive load in the mixed load. When this part has a smooth network environment, the performance improvement is obvious and separate networks would reduce the number of requests running on the original network. These two aspects contribute to the performance improvements. And in the W3, W4, and W5 mixed proportion, the bufferless subnetwork has increased the number of data packages, and compared to the buffered subnetwork, the bufferless counterpart is more likely to incur excessive link usage, which deteriorates the performance of latency-sensitive applications. This is a phenomenon consistent with Figure 10, so the COA was proposed. Hybrid COA has an average performance increase of 21%, as opposed to the buffered one.

Figure 15 shows a deflection rate of 128 threads in the bufferless sub-NoC. The average decrease in the deflection rate is 11%, and port contention rate is 7%. For latency-sensitive application loads, the port contention rate is more pronounced. Figure 16 gives a comparison of power consumption under 128 threads, with an average savings of

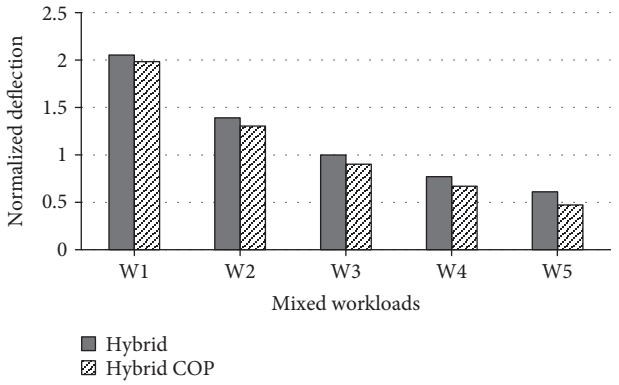


Figure 15: Normalized deflection rate of hybrid loads with 128 threads.

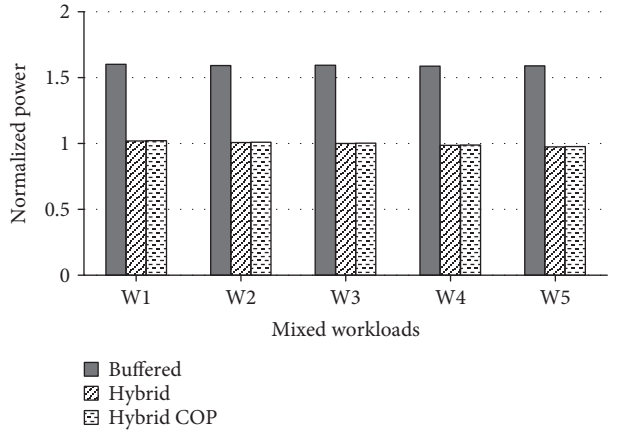


Figure 16: Normalized energy of hybrid loads with 128 threads.

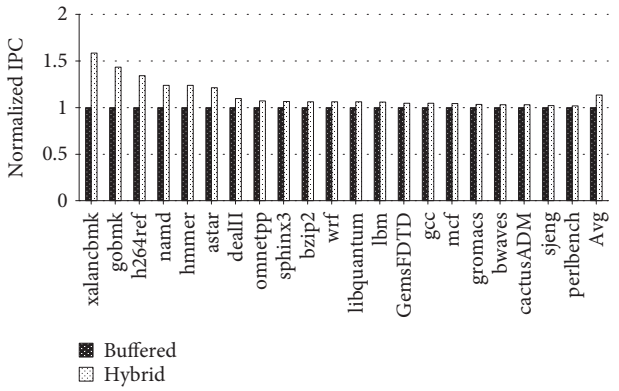


Figure 17: Normalized IPC of 64 threads with one thread per core.

38%. The effect is the same as the power consumption under the 64 threads.

6.3. Analysis on Fairness. This section presents simulation results under big data loads. The processor is of a single thread and two threads per core for the purpose of performance comparison, as shown in Figures 17 and 18. On the mixed network structures, the overall system performance increases by 13.3% on average, with nonlatency-sensitive applications up by 4.63% and latency-sensitive applications

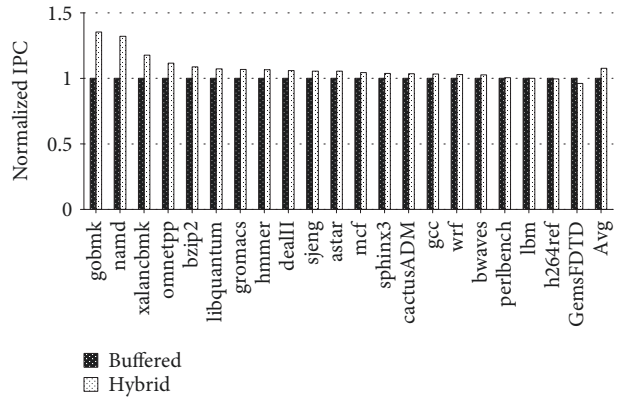


Figure 18: Normalized IPC of 128 threads with two threads per core.

performance up by 30.6% on average. The performances of these two types of applications have been improved. If we exclude the latency-sensitive applications, the performance of the nonlatency-sensitive applications has not been improved significantly. This is because the bandwidth occupied by the latency-sensitive applications is smaller and the nonlatency network is still dominated by its load. The 128 thread results show that the improvement of the whole system performance is up by 7.65%, nonlatency-sensitive applications up by 4.12% on average, and latency-sensitive applications up by 14.7% on average.

7. Conclusion

An on-chip network is an important shared resource for exascale multicore systems which are used in the IoT applications, cognitive computing, and cloud computing. The proper use of it will lead to significantly improved energy efficiency. Due to the chip area and power consumption limit, the NoC has issues related to energy efficiency and average performance. A mono-NoC is designed for common multicore loads, while with the increasing requirements for resources and application characteristics, the severe interference between applications limits the system performance.

To prepare the mono-NoC for big data loads, hybrid NoCs based on the application-aware design are very helpful in improving the quality of service and energy efficiency under massive mixed loading of applications. This paper first proposed a hybrid NoC with a dedicated bufferless NoC and a buffered NoC, as well as an application-aware mechanism to help choose optimal efficient NoC. We proposed a new metric NC_ratio to evaluate the big data load. We examined both the 64-thread and 128-thread systems, and our proposed mechanism shows significant improvements in system performance. Secondly, we proposed a new congestion optimization algorithm for hybrid NoCs. It is implemented by monitoring the congestion status of different NoCs and redistributing the packets in the congesting nodes. Simulation results were presented to show that with the proposed two methods, the energy efficiency of the entire system can be significantly improved.

Data Availability

The (simulation data, integer, and floating) data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported in part by the National Natural Science Foundation of China (Grant no. 61202076), along with other government sponsors. The authors would like to thank the reviewers for their efforts and for providing helpful suggestions that have led to several important improvements in our work. The authors would also like to thank all the staff and students in our laboratory for the helpful discussions.

References

[1] E. B. Sifah, Q. Xia, O. B. O. Agyekum et al., "Chain-based big data access control infrastructure," *The Journal of Supercomputing*, pp. 1–20, 2018.

[2] T. Stepanova, A. Pechenkin, and D. Lavrova, "Ontology-based big data approach to automated penetration testing of large-scale heterogeneous systems," in *SIN '15 Proceedings of the 8th International Conference on Security of Information and Networks*, pp. 142–149, Sochi, Russia, September 2015.

[3] W. Xiang, G. Wang, M. Pickering, and Y. Zhang, "Big video data for light-field-based 3D telemedicine," *IEEE Network*, vol. 30, no. 3, pp. 30–38, 2016.

[4] N. Tanabe, S. Tomimori, M. Takata, and K. Joe, "Character of graph analysis workloads and recommended solutions on future parallel system," in *Algorithms and Architectures for Parallel Processing. ICA3PP 2013*, J. Kołodziej, B. Martino, D. Talia, and K. Xiong, Eds., vol. 8285 of Lecture Notes in Computer Science, pp. 402–415, Springer, Cham, 2013.

[5] J. Vetter and S. Mittal, "Opportunities for nonvolatile memory systems in extreme-scale high-performance computing," *Computing in Science & Engineering*, vol. 17, no. 2, pp. 73–82, 2015.

[6] T. Agerwala, "Exascale computing: the challenges and opportunities in the next decade," in *HPCA - 16 2010 The Sixteenth International Symposium on High-Performance Computer Architecture*, p. 1, Bangalore, India, January 2010.

[7] A. Borghesi, M. Bartolini, M. Lombardi, M. Milano, and L. Benini, "Predictive modeling for job power consumption in HPC systems," in *High Performance Computing: 31st International Conference, ISC High Performance 2016*, J. Kunkel, P. Balaji, and J. Dongarra, Eds., vol. 9697 of Lecture Notes in Computer Science, pp. 181–199, Springer, Frankfurt, Germany, 2016.

[8] A. Kumar, *Intel's New Mesh Architecture: The 'Superhighway' of the Data Center*, IT Peer Network, 2017.

[9] B. Chemli, A. Zitouni, A. Coelho, and R. Velazco, "Design of efficient pipelined router architecture for 3D network on chip," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 7, pp. 188–194, 2017.

[10] Z. Qian, P. Bogdan, C. Tsui, and R. Marculescu, "Performance evaluation of NoC-based multicore systems: from traffic analysis to NoC latency modeling," *ACM Transactions on Design Automation of Electronic Systems*, vol. 21, no. 3, pp. 1–38, 2016.

[11] M. Hamidia, N. Zenati-Henda, H. Belghit, and M. Belhocine, "Markerless tracking using interest window for augmented reality applications," in *2014 International Conference on Multimedia Computing and Systems (ICMCS)*, pp. 20–25, Marrakesh, Morocco, April 2014.

[12] P. P. Valentini, "Natural interface for interactive virtual assembly in augmented reality using leap motion controller," *International Journal on Interactive Design and Manufacturing (IJIDeM)*, pp. 1–9, 2018.

[13] T. Xu, W. Xiang, Q. Guo, and L. Mo, "Mining cloud 3D video data for interactive video services," *Mobile Networks and Applications*, vol. 20, no. 3, pp. 320–327, 2015.

[14] H. Bokhari, H. Javaid, M. Shafique, J. Henkel, and S. Parameswaran, "Supernet: multimode interconnect architecture for manycore chips," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, San Francisco, CA, USA, June 2015.

[15] K. Kanoun, M. Ruggiero, D. Atienza, and M. Schaar, "Low power and scalable many-core architecture for big-data stream computing," in *2014 IEEE Computer Society Annual Symposium on VLSI*, pp. 468–473, Tampa, FL, USA, July 2014.

[16] M. Ramakrishna, P. V. Gratz, and A. Sprintson, "GCA: global congestion awareness for load balance in networks-on-chip," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 7, pp. 2022–2035, 2013.

[17] F. Beneventi, A. Bartolini, C. Cavazzoni, and L. Benini, "Continuous learning of HPC infrastructure models using big data analytics and in-memory processing tools," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pp. 1038–1043, Lausanne, Switzerland, March 2017.

[18] E. Isaac, M. R. Babu, and J. Jose, "Impact of deflection history based priority on adaptive deflection router for mesh NoCs," *Electronic Government*, vol. 13, no. 4, pp. 391–407, 2017.

[19] A. K. Mishra, O. Mutlu, and C. R. Das, "A heterogeneous multiple network-on-chip design: an application-aware approach," in *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–10, Austin, TX, USA, May 2013.

[20] A. K. Mishra, N. Vijaykrishnan, and C. R. Das, "A case for heterogeneous on-chip interconnects for CMPs," in *2011 38th Annual International Symposium on Computer Architecture (ISCA)*, pp. 389–400, San Jose, CA, USA, June 2011.

[21] K. K. Chang, R. Ausavarungnirun, C. Fallin, and O. Mutlu, "HAT: heterogeneous adaptive throttling for on-chip networks," in *2012 IEEE 24th International Symposium on Computer Architecture and High Performance Computing*, pp. 9–18, New York, NY, USA, October 2012.