IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received October 18, 2019, accepted November 28, 2019, date of publication December 13, 2019, date of current version January 21, 2020.

Digital Object Identifier 10.1109/ACCESS.2019.2958945

Exploration on Routing Configuration of HNoC With Intelligent On-Chip Resource Management

JUAN FANG[1], ZEQING CHANG[1], AND DONG LI[2]
[1]Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China
[2]Department of Electrical Engineering and Computer Science, University of California at Merced, Merced, CA 95343, USA

Corresponding author: Juan Fang (fangjuan@bjut.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61202076, and in part by the Beijing Natural Science Foundation under Grant 4192007.

ABSTRACT The Heterogeneous Network-on-Chip (HNoC) integrates CPU cores, Graphic Processing Unit (GPU) cores, last-level-cache and memory controllers. The heterogeneity of this architecture inevitably brings resource contention and energy shortage. In this work, we first study the impact of different capacity of router buffers on communication delay and energy consumption. After comprehensive evaluation with a spectrum of benchmarks with different characteristics, we reveal the implicit tradeoff between performance and energy consumption under different constrains of buffer resource. We further show that intelligently allocating more on-chip network buffer resources to compute-intensive nodes can significantly improve system performance. We introduce a runtime strategy that dynamically allocates network buffers to applications based on application characteristics. Our evaluation shows that our system reduces the communication delay by 55.47% on average, and reduces energy consumption of the system by 21% on average.

INDEX TERMS Buffer storage, energy consumption, intelligent systems, network-on-chip, GPU.

I. INTRODUCTION

With the tremendous development of the semiconductor technology, hundreds of millions of transistors are integrated into one single chip. Nevertheless, it is difficult to increase the overall computing capability of the chip because of the Moore Limitation. The computing dies in the chip need massive communication resources and the energy consumption of the chip has gradually become the major factor restraining the whole performance of the system. Prior efforts have put forward the network-on-chip technology that integrates multiple computing dies into one chip to resolve the problem [1]. It is prevalent to integrate GPU and CPU computing dies in the chip in recent years. Fig. 1 shows a typical CPU-GPU heterogeneous network-on-chip. The GPU dies, CPU dies, last-level cache, and memory controllers, each of which is regarded as individual nodes and connected to a router, build a mesh. Each router has east, west, south, north and local input and output ports. Those ports are configured with buffers. The routers handle communication among nodes [2].

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

To ensure the quality of service and performance of network-on-chip communication, the communication delay among nodes need to be minimized [3]. More routing buffer resources leads to better system performance and less communication delay. However, traditional network-on-chip only integrates CPU computing dies, and cannot meet the increasing demands of those emerging applications for high throughout and parallel computing capability. The characteristics of workloads run by CPU and GPU have significant difference, which leads to unbalanced data communication between interconnected channels of the HNoC. The traditional load balancing strategies evenly allocate buffers to each die. This can cause significant communication delay in HNoC [4]. As a result, the traditional strategies can result in a large amount of idle times on the CPU nodes, which in turns causes high static power consumption. In contrast, the GPU nodes that communicate more frequently than the CPU ones will suffer from serious congestion in the network links.

A. ABBREVIATIONS AND ACRONYMS

In this paper, we propose a novel buffer allocation strategy for the CPU-GPU based HNoC. The strategy strikes an optimal balance between performance, power consumption and chip design cost. We investigate the characteristics of different

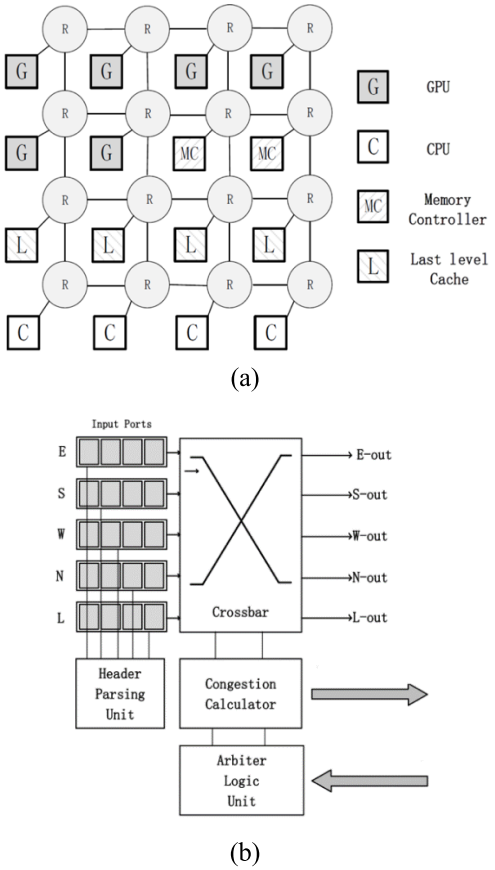

FIGURE 1. (a) CPU-GPU Two-dimension Mesh Heterogeneous Network-on-chip (b) Architecture of an on-chip router.

benchmark applications and analyze the usage of routing buffers in HNoC. Because of the difference between CPUs and GPUs, we show that allocating more buffer resource to busy channel in HNoC can significantly reduce the average communication delay and achieve good performance.

We further investigate how different buffer allocation strategies can affect the communication delay and energy consumption in HNoC. Then, we propose a strategy to statically partition buffers for specific applications. Finally, based on the static partition strategy, we use a virtual channel technology to deal with the imbalance of communication traffic between nodes in the CPU-GPU heterogeneous architecture, to minimize transmission delay and save buffer resources. We combine the virtual channel allocation problem with the queuing theory to construct a mathematical model for on-chip network communication. The model uses the average communication delay of the on-chip network packet as a constraint condition. We use a simulated annealing algorithm to implement the virtual channel allocation strategy. Our method can reduce the buffer resource of the CPU node while meeting the requirements on the buffer sources from the high throughput GPU node. We keep the number of virtual channels always within a reasonable range, thereby reducing the total energy consumption of the system. Finally, we apply the allocation strategy derived from the simulated annealing algorithm to the gem5gpu simulation simulator, and verify the effectiveness of the optimized allocation algorithm.

Our contributions can be summarized as follows:

- An analysis and characterization of CPU-GPU applications;
- An exploration of the impact of different buffer allocation strategies on communication delay and energy consumption with various workloads. We propose a strategy for the HNoC and the computing dies including CPU and GPU. This strategy allocates buffer resources based on the characteristics of the computing cores.
- We performed extensive evaluation of the proposed strategy. Compared with the baseline buffer allocation strategy, our strategy achieves better results in terms of performance and latency in HNoC.
- We apply virtual channel technology to CPU-GPU heterogeneous on-chip network, and combine with the queuing theory to construct a mathematical model to study on-chip network communication. We use the on-chip network average communication delay as a constraint, and use a simulated annealing algorithm to obtain the optimal virtual channel allocation scheme. Compared with the baseline buffer allocation strategy, our proposed intelligent virtual channel allocation strategy can achieve better performance while reducing energy consumption.

B. ORGANIZATION OF THE PAPER

The rest of the paper is organized as follows: Section II presents related work on buffer allocation strategy for NoCs and our motivation. In Section III we describe the characteristic of benchmark application and the design of our static buffer allocation strategy. We show the experimental results of the static allocation scheme and the inspiration for the intelligent virtual channel allocation strategy in section IV. Section V we propose the scheme of HNoC buffer allocation strategy based on virtual channel technology. In Section VI, we show the experiment result of intelligent virtual channel allocation strategy. In section VII, we conclude this paper and discuss future work.

II. BACKGROUND

A. ABBREVIATIONS AND ACRONYMS

In traditional NoCs, buffers reduce the probability of packet loss or path deflection and increase bandwidth utilization. However, buffers also bring some disadvantages. Firstly, buffers consume a lot of energy, including static power consumption and dynamic power consumption. Secondly, management of buffer read and write operations increases the complexity of NoC design. It has been shown that buffers on the TRIPS prototype chip consumes 75% of the NoC area [11]–[13]. Based on this observation, researchers propose a novel bufferless router that eliminates the buffer and design a routing algorithm.

Previous works on buffer saving strategies include [3]–[9]. However, those research mainly focuses on homogeneous

NoCs, by combining bufferless routers and buffered routers to reduce total power consumption. For example, Daya et. al. [10] present a high-performance bufferless mesh NoC that sets up single-cycle virtual express paths dynamically across the chip, allowing deflected packets to go through non-minimal paths width no latency penalty. For a 64 node network, we demonstrate an average 62 percent reduction in latency and an average 1.3 times higher throughput over a baseline bufferless NoC for synthetic traffic patterns.When a large number of packets are transmitted in the NoC and packets compete for a same output port, bufferless routers send packets to deflected output ports. Using this mechanism, some packets cannot be transmitted through an ideal output port directly. Prior research on bufferless NoC design has shown that removing buffers and avoiding performance degradation through novel routing mechanism can reduce network latency and decrease energy consumption at the same time. However, compared to the traditional buffered NoC, even though the bufferless NoC saves a large amount of power, the performance loss caused by data deflection can be non-trivial. It has been shown that bufferless NoCs can only achieve comparable performance to buffered NoCs when the network traffic load is moderate. Once communication pressure increases, packet delay in the network significantly increases, resulting in congested links, decreased overall performance and increased network energy consumption.

On the issue of the rationality of the on-chip network buffer resource allocation strategy, the researchers proposed some novel algorithms. Ahmad [14] studied the impact of the input virtual channel buffer router model on the performance of the on-chip network by changing the packet injection rate and packet length of the on-chip network. The paper simulates the nature of dimension routing in K-ary-n cube mesh topology under different workload conditions to find out the impact of virtual channel throughput and communication delay on Mesh network performance. Experiments show that as the number of virtual channels increases, the throughput and delay of the network will have some optimization space. But when the number of virtual channels reaches a critical value, network performance will reach saturation. Researchers can use this study as a guide to find the optimal number of virtual channels for a particular configuration and workload characteristics.

Wang [3] proposed a novel buffer allocation algorithm with a finite-size buffer and a wormhole exchange on-chip network (NoC). By using the proposed algorithm, higher system performance and lower average communication delay can be achieved compared to the buffer resource average allocation strategy widely used in current NoC designs.

Bao *et al.* [15], proposed a new virtual channel allocation algorithm which is based on the characteristics of the target application workload to customize the virtual channel in the network. Given the communication characteristics of the target application load and the budget of the total number of virtual channels in the on-chip network, the virtual channel can be added one by one to the node port with the highest congestion probability by constructing the congestion probability of each node in the virtual model live network.

Lan and Muthukumar [16] proposed an efficient virtual channel buffer architecture and dynamic virtual channel allocation mechanism to minimize communication delay and local buffer allocation overheadd. The virtual channel buffer architecture and allocation algorithm proposed in the paper can be applied to an on-chip network with buffers and decoupled from the topology of the on-chip network. The architecture can be applied to various traffic patterns and performance evaluations for different load scenarios, compared to existing virtual channel allocation algorithms. Experiments show that the proposed architecture and mechanism have better throughput and smaller chip area than the benchmark allocation algorithm. The limitation of these algorithms is that they are applicable to traditional workloads based on homogeneous computing cores, and do not propose a set of mature best practices in the face of workloads with heterogeneous computing characteristics. HNoC-based workload buffer allocation strategy still have optimized space.

Fang *et al.* [17] proposed a novel application mapping KL_GA algorithm for the mesh-of-tree network topology. The proposed algorithm takes the advantage of both the Kernighan–Lin algorithm and genetic algorithms to reduce the overall communication cost and generates a mapping solution using a KL-based method. In order to avoid the appearance of premature phenomena, researchers next apply a GA-based algorithm to get rid of the population trapped in the local optimum and re-generate a new population.

Fang *et al.* [18] first evaluates the performance and power consumption of a variety of static hot-potato based heterogeneous NoCs with different buffered and bufferless router placements, which is helpful to explore the design space for heterogeneous GPU-CPU interconnection. Then it proposes Unidirectional Flow Control (UFC), a simple credit-based flow control mechanism for heterogeneous NoC in GPU-CPU architectures to control network congestion. UFC can guarantee that there are always unoccupied entries in buffered routers to receive flits coming from adjacent bufferless routers.

Fang *et al.* [19] first evaluates the impact of different capacity of router buffers on communication delay and energy consumption. Then it runs benchmarks to simulate different characteristics of the real-world applications, with the aim to balance performance with energy consumption under buffer resource limitations. The evaluations of HNoC show that when the buffer resources are limited, by allocating more buffer to GPU, the energy consumption decrease by an average of 44.6%, while the performance degradation is negligible.

B. MOTIVATION

To ensure the cooperative work among units of HNoC, the communication quality among units are essential, because the communication characteristics of CPU and GPU are totally different [6], [8]. CPUs are designed to use complex

control logics to allow a single thread to execute instructions in parallel (even for unordered instructions) on the premise of keeping the application executed on demands as a whole. On the contrary, GPUs pursue high throughout and they have a large number of threads. GPUs can shift to other threads and continue to carry out the remaining work when some threads are waiting for the storage to execute access or arithmetic calculation [5]. Traditional buffer allocation strategy leads to congestion in which channels have heavy traffic load. Therefore, such a strategy is not suitable for CPU-GPU heterogeneous networks.

III. DESIGN OF STATIC BUFFER ALLOCATION STRATEGY FOR CPU-GPU BASED HETEROGENEOUS NOC

A. CHARACTERISTIC OF BENCHMARK APPLICATION

CUDA GPGPU and CPU SPEC 2006 benchmarks are used for the experiments in this work. We classify benchmark applications by the evaluation of the number of packets injected into the NoC per kilocycle (PKC).

TABLE 1. GPU benchmark classification.

NVidia CUDA / Rodinia benchmark						
GPU	High (PKC>240)		Medium (240>PKC>100)		Low (PKC<100)	
	Benchmark	PKC	Benchmark	PKC	Benchmark	PKC
	Aligned	504	FDTD3d	174	Backprop	97
	BlackScholes	272	AsyncAPI	142	Hotspot	43
	ScalarProd	247				

TABLE 2. CPU benchmark classification.

SPEC 2006				
CPU	High (PKC>35)		Low (PKC<35)	
	Benchmark	PKC	Benchmark	PKC
	Mcf	74	Calculix	14
	Omnetpp	55	Wrf	11
	Bzip2	41	Gcc	7
	Poveray	37	catusADM	6

Table 1 and Table 2 show the classification results of benchmark applications. We classify the GPU benchmark applications into three types based on the value of PKC. If the PKC is less than 100, the application is considered as the one with a low packet injection rate; if PKC is between 100 and 240, the application is considered as the one with a middle packet injection rate. If PKC is greater than 240, the application is considered as the one with a high packet injection rate. As for CPU applications, an application is regarded as the one with a low packet injection rate, if its PKC is no bigger than 35; the application is regarded as the one with a high packet injection rate, if its PKC is bigger than 35.

B. DESIGN OF ROUTER ARCHITECTURE

The router architecture of our proposed NoC is shown in Fig. 1(b). Each input port has a separate buffer that stores the in-coming data packets before they are transmitted to downstream output port. Each input port buffer can have a different capacity. Considering that in HNoC, CPU node throughput is much smaller than the GPU node, we assume that the link areas with high communication demands are mainly located in the GPU Die. Thus, we allocate more buffer resource for GPU nodes, which means that the chip area of the GPU router is larger than that of the CPU.

Compared with traditional on-chip design, buffered router in static HNoC employ different design methods. Traditional NoC routers use worm-hole routing algorithm, and the granularity of route calculation is data packet. In HNoC, the granularity of route calculation is flit. Regardless of whether a data flit is located at the head, body or tail of the packet, it will carry routing information. Conventionally the granularity of the data injected into NoC by the network interface is flit, packets are not divided into flits at the time of injection, but this is not the case in HNoC. In order to ensure the fairness of data transmission, data flits of a same data packet are injected into a same virtual channel first. If the current virtual channel is full, the remaining data flits can be injected into other virtual channels.

In HNoC, all routers employ Hot Potato routing. In order to ensure that the data flit will not be dropped when the target downstream router has small buffer capacity, data flit transmission will never stop regardless of whether or not the virtual channel of its neighboring router has space left. This will lead to the situation that the number of data flits are larger than the total number of buffers of a router, and we need to employ Hot Potato deflecting routing. At the same time, the original credit-based flow control mechanism is no longer applicable, and when there is no remaining space in virtual channel of target router, neighboring routers are allowed to transmit data flit to that router. The routing process in HNoC is divided into two phases. The first phase is the routing calculation and virtual channel allocation. The second phase is switch allocation and traversal. It takes two clock cycles for a flit to pass through a router if there is no contention.

The deflection of the data flits will cause some flits fail to reach their destination in order. Thus, it is necessary to employ a buffer to store data flits of a packet. If all data flits belonging to a same packet arrive at the destination, the buffer will assemble these data flits into a complete data packet and deliver it to the CPU core, GPU core, last-level cache or memory controller.

IV. EVALUATION OF PROPOSED STATIC BUFFER ALLOCATION STRATEGY

A. EVALUATION METHODS

We choose Gem5-GPU as our evaluation platform [22]. Gem5-GPU is well modularized with a detailed memory module. Gem5-GPU can also faithfully simulate various configurations of CPU-GPU heterogeneous networks-on-chip.

GPUWattch is used to perform NoC power analysis for the power consumption module. The proposed NoC uses the XY deterministic routing algorithm. This is because complex routing algorithms require complex communication protocols and circuit elements. A simple routing algorithms can reduce the complexity of design while the chip area and power consumption are manageable. We employ a 4 × 4 mesh structure as the prior work [20], [21]. The simulated chip contains 4 CPU dies, 6 GPU dies, 4 last-level cache modules and 2 memory controllers, as shown in Fig. 1(a). In the Gem5-GPU simulator, each CPU die runs a CPU benchmark application while all GPU dies execute one benchmark application together, the characteristics of the GPU execution task are determined by the architecture of the Gem5-GPU [22].

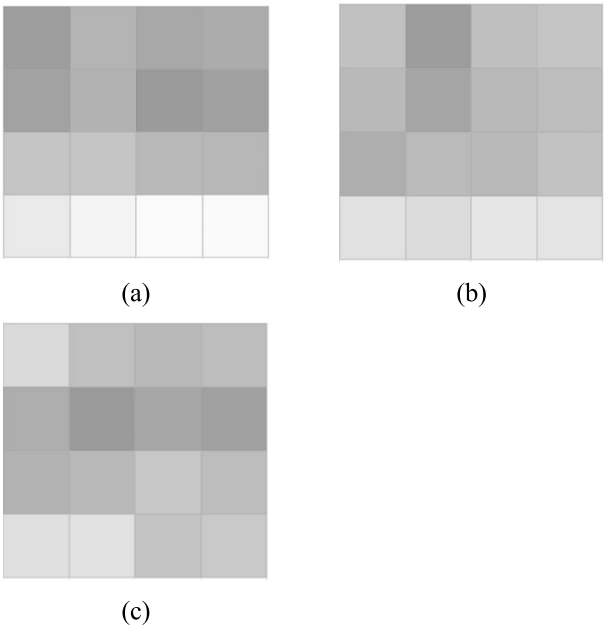

FIGURE 2. Occupancy of HNOC Router Buffer Resource (a) benchmark1 (b) benchmark2 (c) benchmark3.

We carry out three groups of experiments, based on the classification results of benchmarks. One GPU application and four CPU applications are selected as one concurrent workload. In the three groups of experiments, we use different workloads on GPU. We allocate 11× flit size (FS) buffers on average for the input ports of the four channels in east, west, south, and north of each node. The HNoC runs different combinations of benchmark applications, with a heat map drawn according to the average occupancy of the buffer. As shown in Fig. 2, The squares represent the computing dies in the mesh topology illustrated in Fig. 1. The darker color a square has, the higher rate of occupancy the routing buffers at that node have. Fig. 2(a), benchmark 1, shows the result for GPU benchmarks with CPU benchmarks CatusADM, Gcc, Wrf and Calculix. Fig. 2(b), benchmark 2, illustrates result for GPU benchmarks BlackScholes with CPU benchmarks catusADM, Gcc, Poveray, and Bzip2. Fig. 2(c), benchmark 3, shows result for GPU benchmarks FDTD3d with CPU benchmark Mcf, Omnetpp, Bzip2, and Poveray.

B. RESULTS ANALYSIS

Our experiment results show that there is a significant difference in occupancy of input buffers at different nodes in HNoC. Different combinations of benchmarks create different hot spots. The hot spots in Fig. 2(a) are three nodes at the positions (0,2), (0,3), and (2,2). The hot spots in (b) are two nodes at the positions (1,2) and (1,3). The hot spots in (c) are two nodes at the positions (1,2) and (3,2). The link areas with high communication demands are mainly concentrated at the GPU die, the memory controller and the last-level cache.

Because of the heterogeneity of HNoC, the traffic patterns are also different from the traditional NoCs, and show obvious non-uniformity. There is obvious difference in various benchmark applications. We divide all the routers into three types, where the first type connects the CPU die, the second type connects the last-level cache or memory controller, and the third type connects GPU die. The relationship between the PKC of the associated channels and the average communication delay is characterized respectively, when three types of routers configure buffers with different sizes. To make the statistical results more accurate, the experiment ignores the packet transmission within 200 clock cycles of the initial operation stage of the HNoC, since it will take a certain amount of time for the channel communication to reach a stable stage.

According to the results shown in Fig. 3, as the buffer size increases, the average communication delay of the correlated channel declines under different PKCs, but the declination degree varies. The PKC of CPU benchmarks is much smaller than that of GPU benchmarks, thus the impact on the average communication delay of the entire system is very limited when we increase the buffer capacity of the CPU-related communication channel.

For example, for the first type of node, when the PKC is equal to 180, the average delay of the correlated channel is in the range between 478 and 646 cycles. In this situation, there is no significant difference, if the buffer size is bigger than 5 in terms of the PKC. The communication pressures of the second and third type of routing nodes are relatively high. When the PKC increases, the average communication delay can be significantly reduced by increasing the input buffer size. For example, when the PKC is 500, and the input buffer size of GPU dies is 5×FS, the average delay is 12351 cycles. However, when the buffer size is 11×FS, the average communication delay is only 3782 cycles, meaning that the average delay drops by 64.3%. Meanwhile, when the PKC is 500, and the memory controller/last-level cache node buffer size is 5×FS, the average communication delay is 9508 cycles. However, when the memory controller/last-level cache node buffer size is 11×FS, the average packet communication delay is 2782 cycles, meaning that the average communication delay of the system drops by 70.7%.

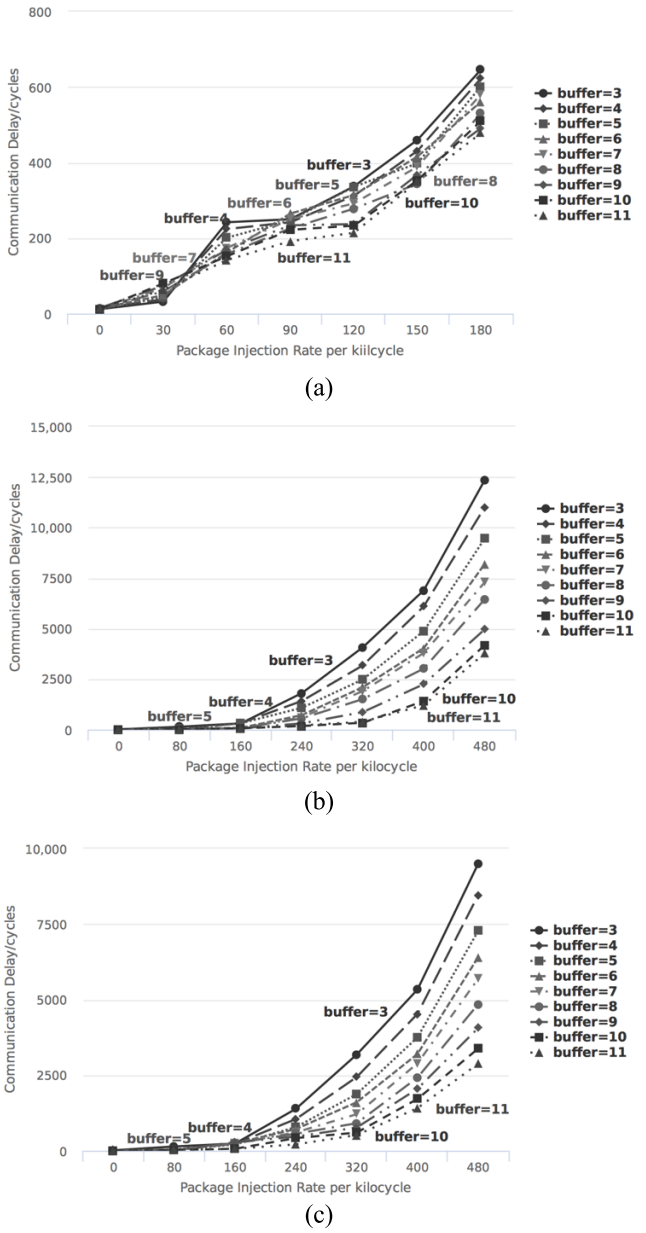

FIGURE 3. Communication delay under different PKCs (a) The PKC of CPU benchmarks (b) The PKC of GPU benchmarks (c) The PKC of memory control/last-level cache benchmarks.

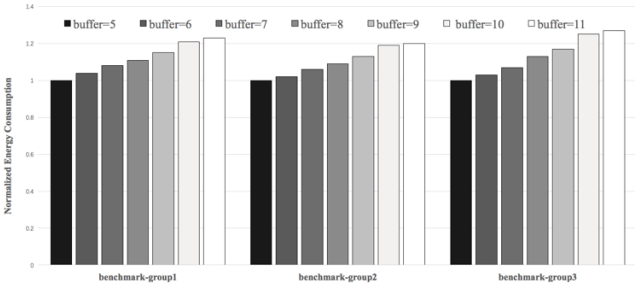

FIGURE 4. Energy consumption with different CPU router buffer size.

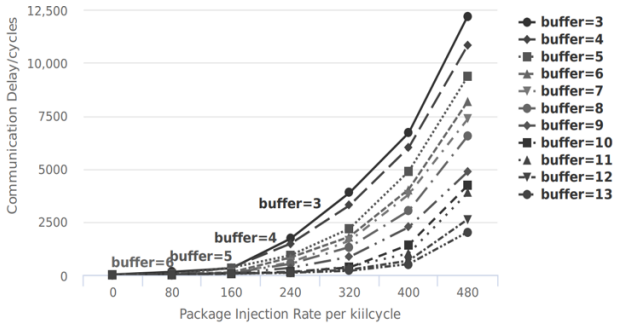

FIGURE 5. The relationship between GPU buffer size and communication delay.

Fig5. shows that there is no significant performance improvement brought by allocating a large number of buffers to the GPU die due to the heterogeneity of the HNoC. When the buffer capacity of the CPU router buffer reaches a critical value, increasing the buffer size of the first type of the computational node mentioned in Table 2 has very little impact on the average communication delay of the system. Because the congested links in the HNoC are mainly in the communication links related to the second and third types of nodes at this time, the average communication delay is maintained at a quite high level due to the severe shortage of buffers connected with these links. The average communication delay of the system drops significantly when we increase the buffer capacity in these two types of node links. This indicates that it is very effective to allocate more buffer resources for links with greater communication demands in the case of limited buffer resources.

This non-uniform buffer allocation strategy can be used to allocate buffer resources more reasonably to the most needed nodes, significantly improving the overall performance of the system. In order to verify our point of view, we allocate buffer resources with different capacities for the first type of nodes (i.e., CPU) and evaluate the energy consumption of the network. The experimental results are shown in Fig. 4. With the increase of the buffer, the energy consumption shows an increasing trend. Additionally, according to the experiment of Fig. 3, we can see that there is a very subtle performance gain by improving the buffer size when the buffer size of the first type of the node exceeds the critical value 5×FS. In order to balance system performance, energy consumption and design cost at the same time, the buffer size of the first type of node is limited to 5×FS in this paper. As a result, the total system energy consumption is reduced by an average of 23.3% compared to when the buffer size is 11×FS. Limiting the buffer size to 5xFS also reduces the chip area, thereby reducing static power consumption and circuit complexity.

In order to further improve the overall performance of the network and decrease the communication delay, we propose to allocate more buffer resources to the heavy communication demanding channels on the basis of total buffer capacity of the HNoC. From Fig. 3 (a) and Fig. 4, we can see that the performance, energy consumption and cost of the system can

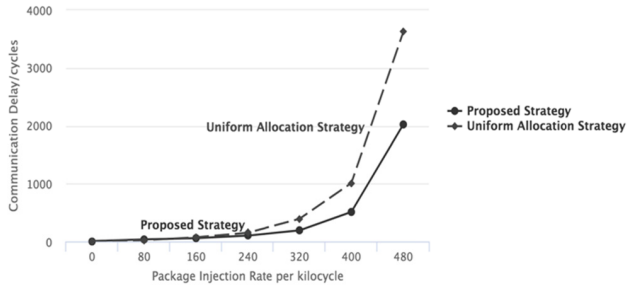

FIGURE 6. The comparison of traditional average buffer.

be kept within a reasonable range at the same time when the first type of the routing node buffer is 5×FS. we set the buffer size of each channel of the second and third type of routers to 13×FS, and that of the first type of router is set to 5×FS. Benchmarks with different combinations are executed and compared with the traditional buffer resource allocation strategy. We measure the communication delay of the system under each configuration. Fig. 5 shows that when the PKC is 480 and GPU/memory controller/last-level cache node buffer is 11×FS, the average packet communication delay of the system is 3794 cycles. When the GPU/memory controller/last-level cache node router buffer size is 13×FS, the communication delay of the system is 2078 cycles, meaning that the average communication delay of the system drops by 44%. Fig. 6 shows the comparison of the traditional buffer allocation strategy with the allocation strategy proposed in this paper. When the PKC is 480, the traditional buffer allocation strategy allocates the buffer with the size of 11×FS for each router. Under this configuration, the average packet communication delay of the system is only 3748 cycles. The average communication delay of the system drops by 44.6% when the strategy proposed in this paper is applied in buffer allocations.

V. DESIGN SCHEME OF BUFFER ALLOCATION STRATEGY BASED ON VIRTUAL CHANNEL TECHNOLOGY FOR CPU-GPU HETEROGENEOUS ON-CHIP NETWORK

A. MOTIVATION

In the CPU-GPU heterogeneous on-chip network, the communication between different nodes is unbalanced due to the characteristics of the CPU and GPU nodes and the characteristics of the workloads running on CPU and GPU. This results in higher buffer usage in some compute node regions and lower buffer utilization in other regions. Unreasonable allocation of on-chip network buffers can result in extremely high energy waste. Although we have proposed a static buffer allocation strategy, it does not have good adaptability and versatility in the face of real world applications. We need to find a more versatile solution. There is existing research on the non-uniform distribution of virtual channel resources. In the existing work [23], the buffer capacity is adjusted based on a greedy algorithm and the bandwidth utilization of the node input port is used as a constraint. The existing work determines the buffer allocation strategy according to the communication characteristics of the nodes in the network on the chip and the surrounding nodes. The limitation of the existing work is that the proposed algorithm only allocates virtual channel resources from the perspective of a single input channel, without considering the communication of each link of the entire system. As a result, the allocation strategy is not very effective. This paper explores the virtual channel algorithm for the CPU-GPU heterogeneous on-chip based on an annealing genetic algorithm.

B. OVERVIEW OF VIRTUAL CHANNEL TECHNOLOGY

The virtual channel allocation problem of CPU-GPU heterogeneous on-chip network is an NP problem. The data exchange between nodes in the system is highly dynamic, and the traffic characteristics and application characteristics of the on-chip network have strong correlation. In this paper, we use the queuing to formulate the virtual channel problem. We establish an analytical model for the CPU-GPU heterogeneous on-chip network communication based on the mesh architecture. Using the proposed model, we can accurately estimate the average communication delay of packets in the network on the chip. On this basis, we use the average communication delay in the on-chip network as a constraint, and use an annealing genetic algorithm to derive the virtual channel allocation strategy of the CPU-GPU heterogeneous on-chip network for applications. Compared with the traditional strategy of allocating buffers, our algorithm can allocate the on-chip network buffer resources more reasonably while ensuring the overall performance of the system, and control the total number of virtual channels in the chip within a reasonable range. We apply this virtual channel allocation strategy to the on-chip network simulator to verify the effectiveness of the strategy.

C. CPU-GPU HETEROGENEOUS VIRTUAL CHANNEL ARCHITECTURE

In traditional on-chip networks, chip designers typically use single-channel buffers to reduce static power and design costs associated with circuit complexity. The disadvantage of this design idea is that in the face of a specific application, the unbalanced communication between the network nodes on the chip will leave a large number of buffers underutilized, resulting in the waste of power consumption and performance degradation. This section uses the virtual channel technology to optimize the power and performance issues caused by the unbalanced communication in the heterogeneous on-chip network.

In the process of designing the chip, if the buffer capacity of the input port is required to be 16-flit sized, the single channel mode shown in Fig.1a can be used. This fact is depicted in Fig. 7. In the figure, 16 buffer units are connected in series a data packet storage. If the virtual channel technology is used, the 16 buffer units can be split. For example, the buffer unit can be connected in a $4 * 4$ or $2 * 8$ manner, and results

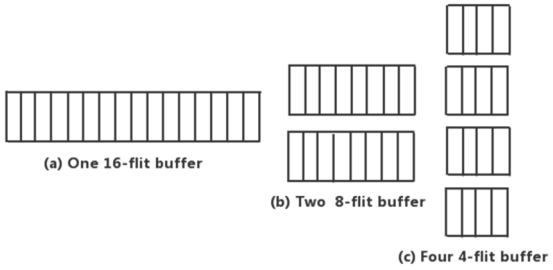

FIGURE 7. Storage arrangment with 16 flits of buffer units.

in the parallel routing channel, shown in Fig. 7(a) Fig. 7(b), Fig. 7(c).

References [24]–[28] studies how to guarantee Qos in HNoCs in order to attain high energy efficiency and performance. An on-chip network using virtual channel technology can improve the quality of service on the on-chip network [29], [30], as shown in Fig. 8. In Node1, when a flit from the package A is waiting for a computing resource, the flit waiting for the resource is temporarily stored in the buffer A. At the same time, the routing node can continue to process the flit from the package B. The traditional on-chip network design usually adopts the single-channel wormhole routing. When an input port is occupied, even if the downstream communication link is idle, the port cannot process flit from two different packets at the same time. If the virtual channel technology is used, the input buffer of the node route can process flits of multiple packets at the same time. This will increase the throughput of the chip and improve the overall performance of the system. Figure 2 shows the difference between single-channel routing and virtual channel routing when processing packets through the buffer.

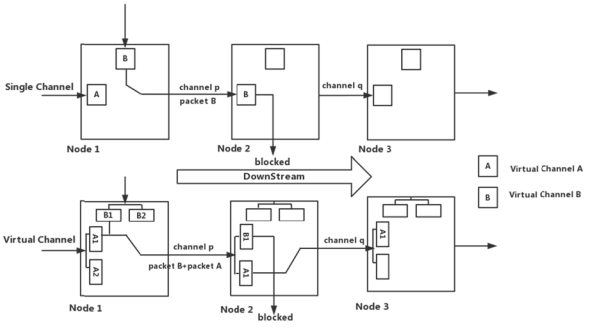

FIGURE 8. An on-chip network using virtual channel technology.

The two technologies shown in Fig. 8 [21] can be analogized to road driving: there is a vehicle A (packet A) that stops at the one-way road and waits for a left turn. Since the road is a one-way street, even if the straight road is not congested, the vehicle B (the packet B) which is behind the vehicle A still cannot go straight. Adding a virtual channel to the on-chip network is analogous to adding a left-turn road. Because of the left turn road, the vehicle B can continue to drive forward over the vehicle A. Adding a left-turn road does not change the fact that the road is a one-way street, that is, adding a virtual channel does not add a physical link, and thus does not increase the bandwidth of the channel.

However, in the field of on-chip network design, when the buffer size is determined, increasing the number of virtual channels will reduce the buffer depth of each channel, causing the microchips from the same packet to be scattered among several routing nodes, which will increase congestion and communication delays in the on-chip network. Moreover, the increase in the number of virtual channels will greatly increase the complexity of the circuit and the chip area, resulting in a large amount of static power consumption. A heterogeneous on-chip network based on GPU and CPU must consider both the overall performance and power consumption of the system, and strike a balance between the two to achieve optimal results. We propose a mathematical model of power consumption based on CPU-GPU heterogeneous architecture, shown as follows:

$$P_{\text{router}} = 3.6 + 1.78\delta_{\text{buffer}} + 6.58\delta_{\text{vc}} + 0.468\delta_{\text{flit}} \quad (1)$$

In Eq. (1), δ_{buffer} is the virtual channel cache depth, δ_{vc} is the number of virtual channels, and δ_{flit} the packet microchip size. It can be seen that the constant term of δ_{vc} is the largest, that is, the number of virtual channels in the CPU-GPU heterogeneous on-chip network is controlled within a reasonable range. This ensures that we can manage the total power consumption of the system while maintain the same performance.

D. MATHEMATICAL MODAL OF VIRTUAL CHANNEL ALLOCATION STRATEGY FOR CPU-GPU HETEROGENEOUS ON-CHIP NETWORK BASED ON MESH ARCHITECTURE

We have a design goal for the virtual channel allocation algorithm for CPU-GPU heterogeneous on-chip network: In the mesh structure, the CPU and GPU with different computing characteristics are guaranteed to have high throughput, and the overall communication delay of the system is degraded. Under the premise, the allocation algorithm proposed in the paper should dynamically allocate the buffer resources for each node, thus minimize the total number of virtual channels in the whole system. Since the computing characteristics of different applications vary greatly, and the communication between the network nodes on the chip and the exchange of data packets are a complex dynamic process, it is difficult to quantify the transmission delay of each clock cycle, so we select on-chip for specific applications. We use the packet average communication delay in the network as a constraint to establish a mathematical model. The mathematical definition of this constraint is shown in Eq. (2):

$$N = \sum_{\forall x,y,z} V_{x,y,z} \quad (2)$$

In Eq. (2), $V_{x,y,z}$ represents the number of virtual channels of the routing node (x, y) in the z direction (Including East, South, west, North and Local).

For a specific application, we use a deterministic routing algorithm under the mesh structure, and the constraints are as shown in Eq. (3):

$$\mathrm{T_N} \leq T_{Normal} \tag{3}$$

In Eq. (3), T_N is the average communication delay of the CPU-GPU heterogeneous on-chip network packet based on virtual channel technology; T_{Normal} the average packet communication delay in the traditional on-chip network. The buffer allocation strategy adopted in the traditional on-chip network is the average allocation buffer strategy. The optimization goal of this paper is min $f_0[\![(\mathrm{N})]\!]$.

When the workload is running in an on-chip network, the communication process in the on-chip network can be treated as the sum of data exchanges between pairs of nodes. Because the workload represents different computing characteristics of the actual production environment, the communication traffic and packet exchange rate between node pairs are different under different workload conditions. Because of the different characteristics of CPU and GPU computing nodes, communication links near different nodes may experience different degrees of congestion. This congestion will bring hot spots to the on-chip network. This high degree of dynamics makes it extremely difficult to accurately calculate the transmission delay in real time. Therefore, we use the communication delay of the on-chip network as a constraint to simulate the communication process of the on-chip network. The queuing theory is a powerful tool for constructing this mathematical model. When the workload and routing algorithm are determined, the communication between the pairs of nodes can be quantified. Therefore, the on-chip network average transmission delay can be expressed as:

$$\mathrm{T_N} = \frac{1}{N_{stream}} \sum_{\mathrm{i}=1}^{\mathrm{N_{stream}}} L_{stream_i} \tag{4}$$

In Eq. (4), N_{stream} represents the number of pairs of communication nodes in the network on the chip; L_{stream_i} represents the transmission delay of communication between the i-th node pair. The calculation phase of L_{stream_i} begins at the moment the packet header enters the transmission link and continues along the on-chip network communication link until it reaches the destination node. Its value consists of the delay time $\mathrm{B_{x,y,z}}$ blocked by the z-direction input buffer of all routing nodes (x, y) passing by the packet and the time T blocked by the packet header node in the source node. We assume that $\mathrm{S_{flit}}$ represents the width of the flit, the time that a flit enters the routing switching unit and passes through the routing node is set to $\mathrm{T_{header_i}}$ $\mathrm{T_{header_i}}$. $\mathrm{N_{hop}}$ represents the number of hops between the source node and the destination node during packet transmission. Based on the above assumptions, Eq. (5) can be obtained:

$$L_{stream_i} = \mathrm{T_b}(\mathrm{N_{hop}} + \mathrm{S_{flit}} - 1) + \sum\nolimits_{x,y} B_{x,y,z} + \mathrm{T_{header_i}} \tag{5}$$

Under the premise that the routing algorithm uses deterministic routing, we need to get the routing blocking time $B_{x,y,z}$ and the congestion time $\mathrm{T_{header_i}}$ of the header flit of the packet in the source node. $B_{x,y,z}$ can be calculated from the congestion probability $\theta_{\mathrm{x,y,z}}$ of the routing node (x, y) in the direction z and the congestion time $\xi_{x,y,z}$. As shown in Eq. (6):

$$\mathrm{B_{x,y,z}} = \theta_{x,y,z}\xi_{x,y,z} \tag{6}$$

The congestion time $\xi_{x,y,z}$ in the virtual channel buffer of the data packet in the z direction of the routing node (x, y) can be calculated by queuing theory, as is shown in Eq. (7):

$$\xi_{x,y,z} = \frac{\lambda_{\mathrm{x,y,z}}}{2\left(1 - \lambda_{\mathrm{x,y,z}} S_{x,y,z}\right)} \left[1 + \frac{\left(S_{x,y,z} - MT_b\right)^2}{S_{x,y,z}^2}\right] \tag{7}$$

In Eq. (7), $\mathrm{S_{x,y,z}}$ and $\lambda_{\mathrm{x,y,z}}$ represent the average service time and arrival rate of the packet in the z direction in the routing node (x, y), respectively.

The calculation of the average package arrival rate $\lambda_{\mathrm{x,y,z}}$ of the routing node (x, y) in the z direction can be obtained by Eq. (8):

$$\lambda_{\mathrm{x,y,z}} = \sum_{\forall \mathrm{j,k}} \sum_{\forall j,k} \alpha_{\mathrm{j,k}} d_{j,k}^{j',k'} \Re\left(\mathrm{j,k,j',k',x,y,z}\right) \tag{8}$$

In Eq. (8), $\alpha_{\mathrm{j,k}}$ represents the packet injection rate of the node (j, k); $d_{j,k}^{j',k'}$ represents the probability that the node (j, k) sends flits to (j', k'). $\Re$ function is the routing function, this paper use deterministic routing as the routing function $\Re$. If the source node (j, k) transmits flits to the (j', k') node, and the flit passes through the input channel in the z direction of the routing node (x, y), then $\Re\left(\mathrm{j,k,j',k',x,y,z}\right)$ is 1; otherwise, the $\Re\left(\mathrm{j,k,j',k',x,y,z}\right)$ is 0.

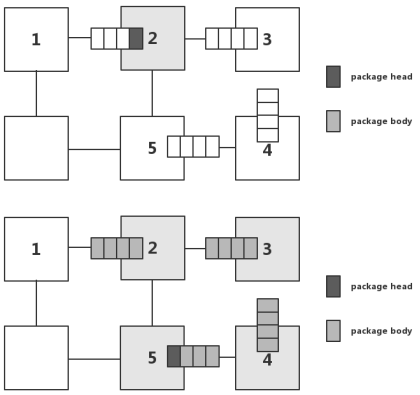

FIGURE 9. Schematic diagram of packet transmission delay.

$S_{x,y,z}$ is the average service time of the z-direction input buffer in the routing node (x, y). In this paper, the on-chip network based on virtual channel technology is used, and the buffer depth is small. The flits from the same data packet are exist in multiple different routing nodes, as shown in Fig. 9. In virtual channel technology based HNoC, the flits of the same data packet will exist in multiple routing nodes at the same time. The calculation of the transmission delay should

start from the packet header enter the transmission link and end when the last flit leaves the link.

Therefore, the average service time of the j-th routing node is as shown in Eq. (9):

$$S_j = MT_b + \sum_{l=j+1}^{j+A_j} B_l \quad j = 1, 2, 3 \tag{9}$$

In Eq. (9), B_l is the buffer blocking delay time when the packet passes through the downstream node l; A_j is the number of downstream router nodes that affect the average service time. Where A_j can be described as Eq. (10). The parameter H represents the virtual channel buffer depth. If the buffer capacity of the input node of the routing node is 16 flit sizes, and four virtual channel each post is used, H is equal to 4; if eight virtual channel each port is used, H is equal to 2.

$$A_j = \begin{cases} \left\lceil \frac{M}{H} \right\rceil, & j \le d - \left\lceil \frac{M}{H} \right\rceil \\ d - j, & \text{Others} \end{cases} \tag{10}$$

$\theta_{x,y,z}$ is the congestion probability of the routing node (x, y) in the z direction. In order to obtain the congestion probability, we should first calculate the packet forwarding probability of each node routing in all directions. This paper chooses the 4∗4 mesh structure as the on-chip network topology. Therefore, in order to describe the packet forwarding probability of each node in each direction, we can It is described as a matrix F, and the matrix subscripts E, S, W, N, and L represent EAST, EAST, SOUTH, WEST, NORTH, LOCAL, respectively.

$$F = \begin{bmatrix} 0 & f_{NE} & f_{NS} & f_{NW} & f_{NL} \\ f_{EN} & 0 & f_{ES} & f_{EW} & f_{EL} \\ f_{SN} & f_{SE} & 0 & f_{SW} & f_{SL} \\ f_{WN} & f_{WE} & f_{WS} & 0 & f_{WL} \\ f_{LN} & f_{LE} & f_{LS} & f_{LW} & 0 \end{bmatrix} \tag{11}$$

For example, f_{NE} located at the position of the matrix F(0, 1) represents the probability that the data packet in the routing node (0, 1) is forwarded from the North direction to the East direction, and the forwarding probability $f_{zz'}$ of the routing node in any direction can be expressed as:

$$f_{zz'} = \frac{\lambda_{zz'}}{\sum_{\forall z'} \lambda_{zz'}} \tag{12}$$

In Eq. (12), $\lambda_{zz'}$ is the packet arrival rate of the routing node input from the z direction and output from the z′ direction. Based on the formula, we can calculate the probability of congestion in the input buffer of the routing node (x, y) of the data packet in the z direction. In order to better present the equation, we use mathematical equations to describe the congestion probability of the input buffer of the routing node (x, y) in the East direction as an example. The congestion of the buffer in the z direction of the routing node (x, y) is shown in Fig. 10.

According to the arbitration of the routing algorithm, we can forward the data flits in the input buffer of the routing

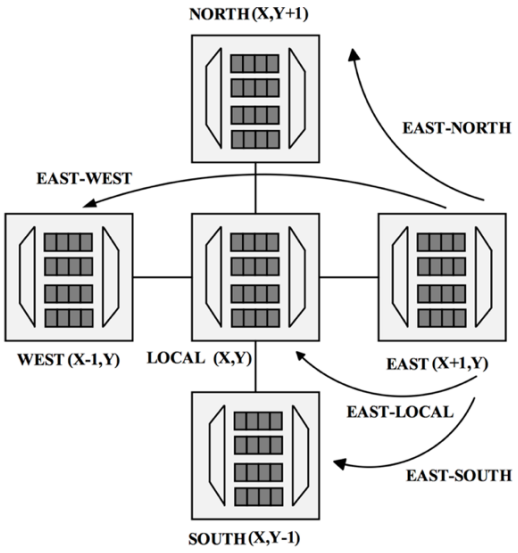

FIGURE 10. Probability of buffer congestion.

node (x, y) in the East direction to the downstream communication link in all directions. Therefore, the congestion probability of the input buffer in the East direction of the routing node (x, y) is the sum of the probability that the input buffer is occupied in each downstream communication link, as shown in Eq. (13):

$$\theta_{x,y,E} = f_{EN} P_{x,y+1,S} + f_{ES} P_{x,y-1,N} + f_{EW} P_{x-1,y,E} + f_{EL} P_{Local_{x,y}} \tag{13}$$

In Eq.(13), $P_{x,y,z}$ is the congestion probability of the routing node (x, y) at the moment the data flits arrive at the input buffer in the z direction. Since this paper uses an on-chip network based on the mesh structure of the virtual channel architecture, the value of $P_{x,y,z}$ is directly related to the allocation mechanism of the virtual channel. Because of the heterogeneity of GPU and CPU, we cannot allocate virtual channel resources evenly. Considering the communication dynamics of the heterogeneity of compute nodes for on-chip network communication, $P_{x,y,z}$ can be considered as the probability that all virtual channels of the routing node (x, y) in the z direction are occupied. Assuming that there are $V_{x,y,z}$ virtual channels in each channel of the routing node, the congestion probability of each virtual channel can be expressed as $\lambda_{x,y,z} S_{x,y,z}$. Hence $P_{x,y,z}$ can be expressed as:

$$P_{x,y,z} = \left(\lambda_{x,y,z} S_{x,y,z}\right)^{V_{x,y,z}} \tag{14}$$

VI. EXPERIMENT RESULTS OF INTELLIGENT VIRTUAL CHANNEL ALLOCATION STRATEGY

A. EVALUATION METHODS

Considering the buffer occupancy of each node when the CPU-GPU heterogeneous on-chip network runs a specific application, we collect the routing node communication parameters during the on-chip network operation at runtime. The running configuration of the Gem5-GPU simulator is

static, in order to better represent the dynamic of the virtual channel allocation strategy. We quantify the application workloads PKC in Gem5-GPU simulator. Base on this, the experiment describes the communication mathematical model of CPU-GPU heterogeneous architecture on-chip network on MATLAB platform.

The basic parameters of the on-chip network communication model are as follows: the on-chip network topology uses a $4 * 4$ mesh structure; The types of compute nodes include GPU and CPU cores; we use the XY deterministic routing algorithm; we assume that the number of virtual channels of each routing node is at least 1, and cannot be greater than 4; The transmission delay L_{stream_i} between pairs of nodes follows the description of Eq. (4).

In order to be consistent with the experimental conditions of the static partitioning buffer scheme described above, we chose the same three benchmark application combinations to simulate to verify the effectiveness of the algorithm. Fig. 2 shows the occupancy of each node buffer under the combination of three benchmark applications. We collect the average throughput and average communication delay of each node during the on-chip network operation. Based on the obtained information, we build an on-chip network communication model on the MATLAB platform. The collected information is used as model parameters, and three sets of experiments are performed under the corresponding parameter sets of different benchmark application combinations. In order to make the statistics more accurate, the experiment ignores the packet transmission process of the initial 200 clock cycles of HNoC operations.

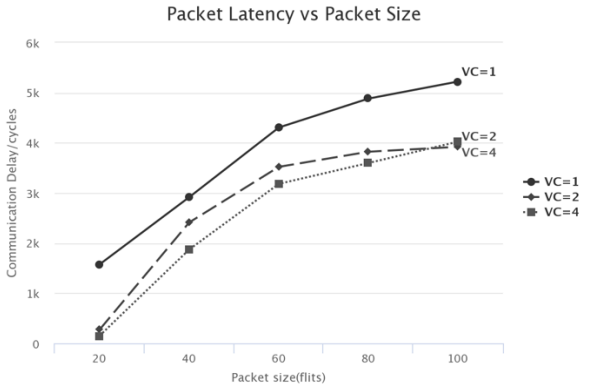

FIGURE 11. The relationship between packet latency and packet size.

B. RESULT ANALYSIS

The experiment first simulates the variation of the average communication delay of heterogeneous on-chip network nodes under different packet lengths and virtual channel numbers, as shown in Fig. 11. We assigned 1, 2, and 4 virtual channels to all nodes in the network on the chip, and set the packet length to 20, 40, and 60. On this basis, we observe the change of communication delay under the condition of different virtual channel numbers. This configuration allows us to observe the impact of different combinations of packet length and number of data channels on CPU and GPU heterogeneous on-chip network performance. In order to ensure the consistency of the experiment, we set the on-chip network packet injection rate to 350.

As shown in Fig. 11, when the number of virtual channels is 1, and the packet length is 20 flit sizes, the average communication delay of the CPU-GPU heterogeneous on-chip network is 1629. As the length of the data packet increases, the average communication delay of the on-chip network increases accordingly. When the packet length is 100-flit sized, the average communication delay of the on-chip network reaches 5223, which significantly affects the performance of the on-chip network. When the number of virtual channels is 2, as the length of the data packet increases, the average communication delay of the on-chip network increases rapidly; When the packet length is 60-flit sized, the average network communication delay grows slowly. As shown in Fig. 11, when the packet length is 100-flit sized, the average communication delay is only increased by 11.4%, compared to when the packet length is 60-fit. When the number of virtual channels is 4, the average communication delay of the on-chip network increases rapidly with the increase of the packet length. When the packet length reaches 60, the average communication delay grows slowly; When the packet length is 100-flit, the average communication delay is only 12.1% higher than when the packet length is 60-flit. We can see that in the case that the number of virtual channels in the CPU-GPU heterogeneous on-chip network is determined, as the length of the data packet increases, the average communication delay of the on-chip network continuously increases to a critical point. When the average communication delay reaches the critical point, increasing the number of virtual channels does not significantly reduce the on-chip network average communication delay.

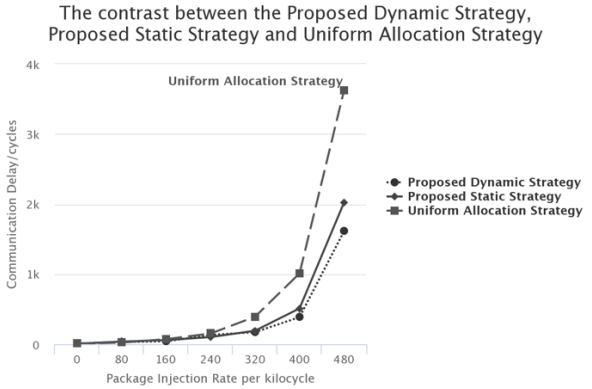

FIGURE 12. Different buffer allocation strategy in communication delay.

The experiment selects the three application combinations in Fig. 2 as the workload, and calculates the average packet communication delay in the CPU-GPU heterogeneous on-chip network at different packet injection rate levels. The experimental results are compared with the static partitioning

strategy and the traditional buffer average allocation strategy. The experimental results are shown in Fig. 12.

When the PKC is less than 320, there is no significant gap between the average packet communication delay of the dynamic allocation virtual channel strategy and the static allocation buffer allocation strategy, and even the average packet communication delay of the static allocation buffer allocation strategy is slightly lower than the dynamic allocation.

The reason for this phenomenon is that when the buffer is allocated statically, the buffer capacity of the node may be larger than the dynamic one, thus the average packet communication delay is maintained at a low level. As the on-chip network packet injection rate increases, communication hotspots appear in the CPU-GPU heterogeneous on-chip network. The communication links related to GPU nodes and MCs are congested. Dynamic allocation of virtual channel policies can allocate buffer resources more reasonably, so the buffer utilization can be improved. As shown in Fig. 12, compared with the static allocation buffer allocation strategy, the dynamic allocation buffer strategy proposed in this paper can reduce the average packet communication delay in the system by 20.23%. Compared with the traditional average allocation buffer strategy, the average packet communication delay can be reduced by 55.47%.

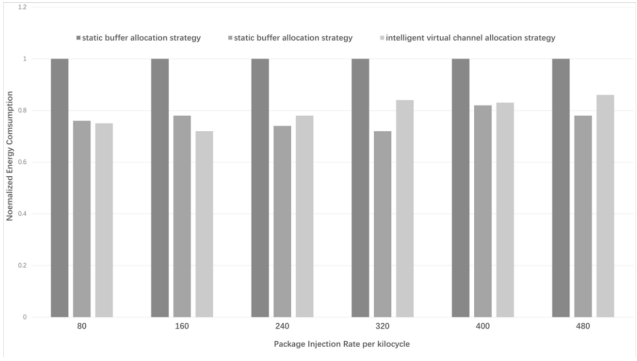

FIGURE 13. Different buffer allocation strategy in energy consumption.

As shown in Fig. 13, the static strategy and dynamic virtual channel allocation strategy proposed in this paper have a declining trend in total system energy consumption when the packet injection rate varies. As the packet injection rate increases, the total system energy consumption of the dynamic virtual channel allocation strategy increases. This is because the utilization of the virtual channel is low when the packet injection rate is low. At this time, under the premise of ensuring the performance of the dynamic virtual channel allocation strategy, the energy consumption is reduced by reducing the number of virtual channels. Compared with the traditional buffer average allocation strategy, the dynamic virtual channel allocation strategy reduces the total system energy consumption by 21% on average.

Compared with the chip architectures commonly used in the industry, the logic components selected in the experiment did not increase in complexity. However, due to the limitations of the experimental environment, the experiment did not provide a detailed comparison between the circuit overhead and the chip architectures commonly used in the industry.

VII. CONCLUSION

In this work, we show that intelligently allocating buffers to the HNoC nodes with different features has good potential for improving system performance and decreasing energy consumption. We first propose a static buffer allocation strategy, and use this as a starting point to propose a method for virtual channel allocation using an intelligent algorithm on HNoC. Specifically, the strategy of dynamically allocating buffer resources can achieve a balance between chip performance and power consumption under different on-chip network traffic characteristics. Our intelligent virtual channel allocation algorithm is shown to be able to effectively decrease communication delay by 55.47% and reduce total energy consumption of the system by 21% on average. The chip architecture and intelligent algorithm involved in this paper have good scalability, which can guide the research direction of larger scale on-chip network.

ACKNOWLEDGMENT

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. They would also like to thank all teachers and students in their laboratory for helpful discussions.

REFERENCES

[1] P. Bhamidipati and A. Karanth, "RETUNES: Reliable and energy-efficient network-on-chip architecture," in *Proc. IEEE 36th Int. Conf. Comput. Design (ICCD)*, Oct. 2018, pp. 488–495.
[2] S. Kundu and S. Chattopadhyay, *Network-on-Chip: The Next Generation of System-on-Chip Integration*. Boca Raton, FL, USA: CRC Press, 2018.
[3] L.-W. Wang, "Optimal buffering resources allocation of on-chip networks with finite buffers," in *Proc. 4th Int. Conf. Intell. Netw. Intell. Syst.*, Nov. 2011, pp. 113–116.
[4] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C.-C. Miao, J. Brown, and A. Agarwal, "On-chip interconnection architecture of the tile processor," *IEEE Micro*, vol. 27, no. 5, pp. 15–31, Sep. 2007.
[5] K. Ma, X. Li, W. Chen, C. Zhang, and X. Wang, "GreenGPU: A holistic approach to energy efficiency in GPU-CPU heterogeneous architectures," in *Proc. 41st Int. Conf. Parallel Process.*, Sep. 2012, pp. 48–57.
[6] J. Zhan, O. Kayiran, G. H. Loh, C. R. Das, and Y. Xie, "OSCAR: Orchestrating STT-RAM cache traffic for heterogeneous CPU-GPU architectures," in *Proc. 49th Annu. IEEE/ACM Int. Symp. Microarchit. (MICRO)*, Oct. 2016, pp. 1–13.
[7] R. James, J. Jose, and J. K. Antony, "Smart port allocation for adaptive NoC routers," in *Proc. 28th Int. Conf. VLSI Design*, Jan. 2015, pp. 475–480.
[8] M. Opoku Agyeman, W. Zong, A. Yakovlev, K.-F. Tong, and T. Mak, "Extending the performance of hybrid NoCs beyond the limitations of network heterogeneity," *J. Low Power Electron. Appl.*, vol. 7, no. 2, p. 8, Apr. 2017.
[9] H. Zhao, M. Kandemir, W. Ding, and M. J. Irwin, "Exploring heterogeneous NoC design space," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2011, pp. 787–793.
[10] B. K. Daya, L.-S. Peh, and A. P. Chandrakasan, "Towards high-performance bufferless NoCs with SCEPTER," *IEEE Comput. Archit. Lett.*, vol. 15, no. 1, pp. 62–65, Jan. 2016.

[11] P. Gratz, C. Kim, R. Mcdonald, S. W. Keckler, and D. Burger, "Implementation and evaluation of on-chip network architectures," in *Proc. Int. Conf. Comput. Design*, Oct. 2006, pp. 477–484.
[12] C. Fallin, C. Craik, and O. Mutlu, "CHIPPER: A low-complexity bufferless deflection router," in *Proc. IEEE 17th Int. Symp. High Perform. Comput. Archit.*, Feb. 2011, pp. 144–155.
[13] C.-K. Hsu, K.-L. Tsai, J.-F. Jheng, S.-J. Ruan, and C.-A. Shen, "A low power detection routing method for bufferless NoC," in *Proc. Int. Symp. Qual. Electron. Design (ISQED)*, Mar. 2013, pp. 364–367.
[14] A. Ahmad, "Investigating the effect of virtual channel on the performance of network-on-chip," M.S. thesis, Dept. Elect. Comput. Eng., Sarhad Univ. Sci. Inf. Technol., Peshawar, Pakistan, 2017.
[15] D. Bao, X. Li, and Y. Xin, "A virtual channel allocation algorithm for NoC," in *Proc. Int. Conf. Machine Learn. Intell. Commun.* Cham, Switzerland: Springer, 2017, pp. 333–342.
[16] Y. L. Lan and V. Muthukumar, "Efficient virtual channel allocator for NoC router micro-architecture," in *Proc. 30th IEEE Int. System-on-Chip Conf. (SOCC)*, Sep. 2017, pp. 169–174.
[17] J. Fang, L. Yu, S. Liu, J. Lu, and T. Chen, "KL_GA: An application mapping algorithm for mesh-of-tree (MoT) architecture in network-on-chip design," *J. Supercomput.*, vol. 71, no. 11, pp. 4056–4071, Nov. 2015.
[18] J. Fang, Z.-Y. Leng, S.-T. Liu, Z.-C. Yao, and X.-F. Sui, "Exploring heterogeneous NoC design space in heterogeneous GPU-CPU architectures," *J. Comput. Sci. Technol.*, vol. 30, no. 1, pp. 74–83, Jan. 2015.
[19] J. Fang, Z. Chang, Y. Cheng, and H. Zhao, "Exploration on routing configuration of HNoC with reasonable energy consumption," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2018, pp. 744–749.
[20] T. T. Le, R. Ning, D. Zhao, H. Wu, and M. Bayoumi, "Optimizing the heterogeneous network on-chip design in manycore architectures," in *Proc. 30th IEEE Int. System-on-Chip Conf. (SOCC)*, Sep. 2017, pp. 184–189.
[21] W. J. Dally and B. P. Towles, *Principles and Practices of Interconnection Networks*. Amsterdam, The Netherlands: Elsevier, 2004.
[22] J. Power, J. Hestness, M. S. Orr, M. D. Hill, and D. A. Wood, "Gem5-gpu: A heterogeneous CPU-GPU simulator," *IEEE Comput. Archit. Lett.*, vol. 14, no. 1, pp. 34–36, Jan. 2015.
[23] A. El-Naggar, A. Medhat, B. Al-Abassy, E. Massoud, H. Ibrahim, M. Khamis, and A. Shalaby, "Performance evaluation of virtual channel flow control in centralized and distributed networks for system on chip," in *Proc. 29th Int. Conf. Microelectron. (ICM)*, Dec. 2017, pp. 1–4.
[24] J. Yin, O. Kayiran, M. Poremba, N. E. Jerger, and G. H. Loh, "Efficient synthetic traffic models for large, complex SoCs," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Mar. 2016, pp. 297–308.
[25] M. K. Jeong, M. Erez, C. Sudanthi, and N. Paver, "A QoS-aware memory controller for dynamically balancing GPU and CPU bandwidth use in an MPSoC," in *Proc. 49th Annu. Design Autom. Conf.-DAC*, 2012, pp. 850–855.
[26] H. Usui, L. Subramanian, K. Chang, and O. Mutlu, "Squash: Simple qos-aware high-performance memory scheduler for heterogeneous systems with hardware accelerators," 2015, *arXiv:1505.07502*. [Online]. Available: https://arxiv.org/abs/1505.07502
[27] D. Shingari, A. Arunkumar, and C.-J. Wu, "Characterization and throttling-based mitigation of memory interference for heterogeneous smartphones," in *Proc. IEEE Int. Symp. Workload Characterization*, Oct. 2015, pp. 22–33.
[28] J. Lee, S. Li, H. Kim, and S. Yalamanchili, "Design space exploration of on-chip ring interconnection for a CPU-GPU heterogeneous architecture," *J. Parallel Distrib. Comput.*, vol. 73, no. 12, pp. 1525–1538, Dec. 2013.
[29] E. Krimer, I. Keslassy, A. Kolodny, I. Walter, and M. Erez, "Static timing analysis for modeling QoS in networks-on-chip," *J. Parallel Distrib. Comput.*, vol. 71, no. 5, pp. 687–699, May 2011.
[30] B. Li, L. Zhao, R. Iyer, L.-S. Peh, M. Leddige, M. Espig, S. E. Lee, and D. Newell, "CoQoS: Coordinating QoS-aware shared resources in NoC-based SoCs," *J. Parallel Distrib. Comput.*, vol. 71, no. 5, pp. 700–713, May 2011.

JUAN FANG received the M.S. degree from the Jilin University of Technology, Changchun, China, in 1997, and the Ph.D. degree from the College of Computer Science, Beijing University of Technology, Beijing, China, in 2005. In 1997, she joined the College of Computer Science, Beijing University of Technology. Since 2015, she has been a Professor with the Beijing University of Technology. Her research interests include high performance computing, edge computing, and big data technology.

ZEQING CHANG received the M.S. degree from the Beijing University of Technology, Beijing, China, in 2019. His main research interest includes computer architecture.

DONG LI received the Ph.D. degree in computer science from Virginia Tech. From 2011 to 2014, he was a Research Scientist with the Oak Ridge National Laboratory (ORNL). He has been an Assistant Professor with the Department of Electrical Engineering and Computer Science, University of California at Merced (UC Merced), Merced, CA, USA, since 2015. He is the Director of the Parallel Architecture, System, and Algorithm Laboratory (PASA). He is also the Lead PI of the NVIDIA CUDA Research Center, UC Merced.