Designing Scalable Hybrid Wireless NoC for GPGPUs

Hui Zhao[1], Xianwei Cheng[1], Saraju Mohanty[1], and Juan Fang[2]
[1]Department of Computer Science and Engineering, University of North Texas
hui.zhao@unt.edu, xianweicheng@my.unt.edu, saraju.mohanty@unt.edu
[2] Department of Information Technology, Beijing University of Technology
fangjuan@bjut.edu

***Abstract*—**Data communication in GPU systems exhibits asymmetric patterns that create congestion hotspots. Due to their large number of cores and big die sizes, GPUs also demand highly scalable NoC designs. In this work, we propose hybrid NoC architectures that employ on-chip integrated antennas to build overlaid wireless networks on top of conventional metal/dielectric-based networks. We use low-power high-bandwidth wireless links as express channels to transmit long distance packets and use metal links to deliver local packets. The hybrid architecture can effectively alleviate congestion near traffic hotspots and improve the throughput and scalability of GPU NoCs. We propose solutions for design challenges in such hybrid NoC architectures, such as MAC protocol, router microarchitecture, load balancing and deadlock free routing. To efficiently utilize on-chip wireless bandwidth, we also propose a novel scheme that adaptively allocates bandwidth to wireless channels based on their usage needs. Our evaluation results show that for a GPU with 256 cores, the proposed hybrid architecture can improve performance by 2.4 times on average.

I. Introduction

Recently GPU accelerated heterogeneous computing is becoming an attractive addition to high performance computing systems due to GPU's ability to launch massive parallel threads at same time. On-chip network need to be able to sustain high data communication throughput in order to reduce the idle time of GPU cores, in order to fully exploit GPU's parallel processing capability. However, compared with CMP based NoCs, design of NoC for GPUs is still at its infancy except for a handful of work [4], [6]–[10].

GPU data traffic has a very different pattern from the CMPs. Many GPU cores send read or write requests to a few memory controllers and receive reply messages from those controllers. The traffic load is imbalanced with reply messages taking up around 70% of total traffic. Hence, GPU NoC has a many-to-few-to-many traffic pattern which generates traffic bottlenecks around the memory controllers. If data generated by memory is not quickly moved away from the memory controllers, network throughput will severely suffer from the resulted congestion hotspots. Another challenge in GPU NoC design is network scalability. GPUs have much larger die sizes compared with CMPs which demands highly scalable network design. Conventional metal/dielectric-based networks do not scale to thousands of cores in GPUs. For example, mesh networks are widely applied to GPU on-chip networks, due to their regular topology and simpler design in both router architecture and routing algorithms. However, hop counts in mesh networks increase significantly in large scale networks which greatly compromises the GPU system performance.

Recent breakthroughs in semiconductor integration technology have enabled new NoC design methods, such as nano-photonic and wireless NoCs, which have the advantage of high bandwidth, speed of light and low power consumption. There have been plenty of research applying these enabling technologies to design CMP based NoCs [11]–[16]. However, only a few of prior works target on GPUs [21], [22].

In this work, we explore the design space by employing wireless on-chip antennas to build hybrid NoC for GPUs. We made the following contributions:
(1) We proposed hybrid NoC architectures that use wireless links to build an overlaid network on top of the wired network. Our proposed design method can not only alleviate congestions near network bottlenecks but also improve network scalability.
(2) We investigated critical design issues in building the hybrid NoCs and propose solutions to address these issues, such as MAC protocol, router micro-architecture and deadlock free routing algorithms.
(3) By taking advantage of the reconfigurability of wireless networks, we developed a scheme to adaptively allocate bandwidth to wireless channels. Our scheme can effectively improve wireless bandwidth utilization.

II. Motivation

GPU NoCs are usually implemented as two separate networks to avoid protocol deadlock. Request messages are sent from many shader cores to a few memory controllers (MCs) in the request network and MCs send reply messages to shader cores using the reply network. Thus the request network exhibits a many-to-few traffic pattern and the reply network has a few-to-many traffic pattern. Traffic load of the two networks is unbalanced, with the reply network carrying most of the data packets. To quantitatively evaluate this load imbalance,

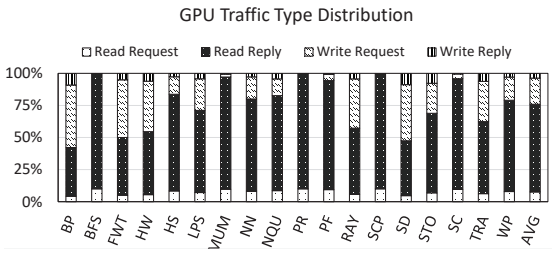
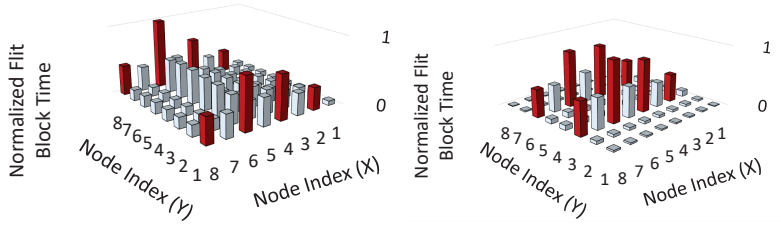


Fig. 1: GPGPU packet type distribution.



(a) Edge MC Placement (b) Distributed MC Placement

Fig. 2: Flit stall time in all routers for workload WP. Red colored bars represent stall time for MC connected routers.

we analyzed the distribution of packet types in Figure 1. It can be observed that about 75% of data is transmitted in the reply network. For some benchmarks, such as *BFS, MUM and PR*, the reply network accounts for more than 90% of all traffic. This is because there are much more read messages than write messages and read reply messages have much larger payload than other types of messages. It is obvious that improving the design of reply network will lead to more performance gains in the overall NoC. This has already been demonstrated by prior works [7], [10]. In this work, our target is to enhance the reply network design.

Due to its few-to-many pattern, traffic inside the reply network is also asymmetric. All packets are injected through the few routers connected with MCs and these routers become the network bottleneck. We evaluated the situation of router congestions in the reply network using workload *WP*. We use flit stall time, i.e., the total amount of time flits are blocked in a router's buffers as a measurement for congestion. It has been shown that MC placements affects the memory-processor traffic flow [4], so we experiment with two typical MC placements: Edge placement and Distributed placement. Edge placement has the advantage of simpler physical design and manufacturing process because the irregular MC nodes are placed at the border of the chip. Distributed placement increases design and routing complexity but has less biased access distance since MCs are located closer to the chip center. As can be observed in Figure 2, the most congested routers are those connected to MCs and their stall time can be x100 times larger than other routers. Routers close to MCs also tend to be congested and congestion gets alleviated as a router's distance from the MC increases. Those routers connected to MCs create network bottlenecks and negatively impact network performance much more than other routers. This is because they block new packets from being injected into the network, even though many other routers are not congested at all. In this work, we propose a solution to reduce network bottleneck by building an overlaid global network on top of the baseline network. When packets are injected from memory, global traffic is immediately directed into an overlaid high speed network and local traffic still goes through the relatively show baseline network.

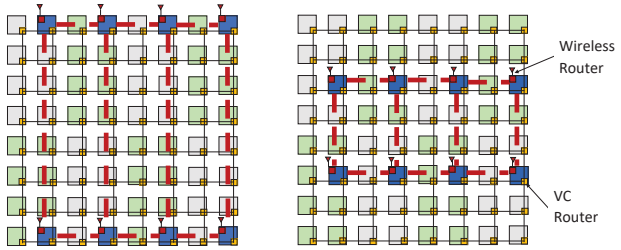
Another critical challenge in designing GPU NoCs is the network scalability. Compared with CMPs, GPUs have much larger die size because they consists of thousands of processing cores. For example, the die size of Intel Core i7 is 246 mm^2, while NVIDIA Pascal GPU has a die size of 610 mm^2. It is expected GPU chip sizes will keep growing as more cores are integrated onto a chip. In conventional NoCs, routers are connected with short metal wires and such design cannot scale with large chip sizes because their performance degrades significantly as hop counts get larger.

Concentration techniques have been proposed to use long metal wires as express channels to connect far apart routers [5]. However, speed of metal wire decreases exponentially as wire length increases [14]. Recent breakthroughs in silicon integrated antennas has opened up new opportunities in designing scalable NoCs. Wireless links have the advantage of ultra-low power, long range communication without speed degradation and reduced wiring overhead. There have been several prior works that explore wireless NoC design for CMPs [12]–[15], but designing wireless NoC for GPUs has yet been well examined. In this work, we propose to build hybrid wireless NoC to reduce GPU traffic bottlenecks and tackle the scalability problem at the same time.

III. Designing Hybrid Wirelss NoC for GPGPUs

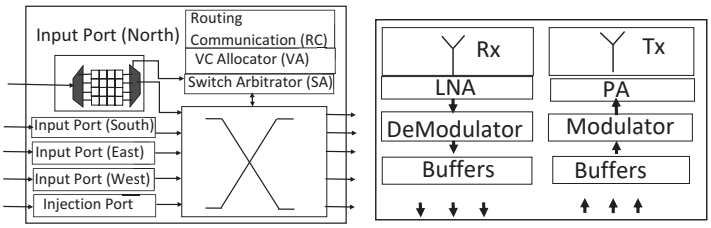
A. Hybrid Wireless NoC Architecture

Our proposed reply network consists two sub-networks: an underlying wired network and an overlaid wireless network. Figure 3 illustrated examples of the NoC architecture with both Edge and Distributed MC placements. The underlying network uses mesh topology and is divided into small clusters, with one memory controller mapped to each cluster. Cluster size is decided by the total number of routers and number of MCs. For example, there are 64 routers and 8 MCs in our example, so the cluster size is 8. The interleaved green and white regions represent clusters and the nodes with blue color represent MCs. Every MC is directly connected to a wireless router and all these wireless routers are connected as an overlaid mesh network. When a packet is injected from a MC, we first decide which network to enter based on the distance to destination. If the number of hops to the destination is below a predefined threshold, the packet will enter the wired network. Otherwise, the packet will be directed to the wireless network and get transmitted to its destination cluster. Then the packet will be ejected from the wireless network into the wired network and continue to reach its destination. Our topology has two differences from conventional concentrated mesh: firstly, the upper level routers (i.e. wireless routers) are not placed at the center of a cluster but MCs; secondly, conventional upper



(a) Edge MC Placement (b) Distributed MC placement

Fig. 3: Topology of the proposed hybrid wireless NoC with different MC placements.



(a) Wired Router (b) Wireless Router

Fig. 4: Major component of wired and wireless routers.

level routers are connected to all routers in the cluster and thus have high radix and larger cost. Each of our wireless router is connected to the MC node only. Our design incurs lower cost by taking advantage of GPU traffic pattern.

All shader cores are connected with wired router only and each MC is connected with both a wired and a wireless router. Our wired routers employs conventional 5-port virtual channel design which is illustrated in Figure 4(a). These routers use look ahead routing and has two pipeline stages. Figure 4(b) shows major components in one wireless channel which has two stacks: physical layer and MAC layer. The physical layer consists of transceivers, modulator/demodulators and buffers. Transceivers use oscillators to generate different carrier frequencies for data modulation/demodulation. Power amplifiers (PA) and low-noise amplifiers (LNA) are used to amplify received or transmitted signals. The MAC layer contains Medium Access Control (MAC) modules that determine when data should be transmitted without causing collisions. Interconnection between the wired and wireless routers is implemented at each MC's network interface. We design control logics in the MC's network interface to handle packet injection/ejection as well as traffic flow coordination between the two networks.

We employ wireless links as express channels to alleviate congestion near memory controllers and reduce network bottleneck. However, our experiments show if all packets with long distant destination are sent to the wireless network, the wireless network will get congested. We need admission control to balance traffic between the two networks. We employ two policies: (1) before a memory controller injects a new packet to the wireless network, it first check the injecting queue length of that router. If the length is above a preset threshold (L_{th}), the packet will be injected to wired network instead. (2) If a packet is blocked in a wireless router for a time longer than a threshold value (W_{th}), the packet will be ejected to the wired network. We use these policies to balance the traffic between the two networks in order to fully utilize the bandwidth of both networks. We use experiments to retrieve these two threshold values and these values are adjustable based on run time situation.

In wireless networks, channels are usually shared by multiple users to improve bandwidth usage. It is important to have a Medium Access Control (MAC) mechanism to avoid collisions. There are several types of MAC developed such as Time Division Multiple Access (TDMA), Frequency Division Multiple Access (FDMA) and Code Division Multi Access (CDMA). We choose token passing as our MAC control mechanism which is a type of TDMA protocol. This is because token based MAC is simple to implement and does not require central control which makes it a better choice for on-chip networks. In this protocol, there is only one token flit circulating among the routers that share one wireless channel. All sharers of a wireless channel are connected by a token ring and a router can transmit data on the shared channel only when it processes the token. In our design, we use a separate high speed control network to pass the token flits.

In our wired and wireless mesh networks, we employ deadlock free routing algorithms to route packets. However, we need to guarantee that there is no deadlock created when packets traverse between these two networks. Deadlocks are created when two packets hold resource (usually buffers) needed by the other packet and are waiting for each other to move forward in a loop. We enforce the following policy to avoid deadlock: a packet can only move from wireless network to wired network, but not in the opposite direction. Once a packet travels on the wired network, it will never be directed to the wireless network. Under this policy, a packet traveling on the wireless network can be blocked by a packet traveling on a wired network when it is ejected from the wireless network. However, a packet traveling on the wired network will never be blocked by a packet on the wireless network. Therefore, no loop can be formed between the wireless and wired network and our network is deadlock free. Our policy works similar to the turn model that avoids formation of loops by forbidding certain turns. Our network is also livelock free since all packets will finally be injected to the underlying network (due to W_{th}) and that network is livelock free.

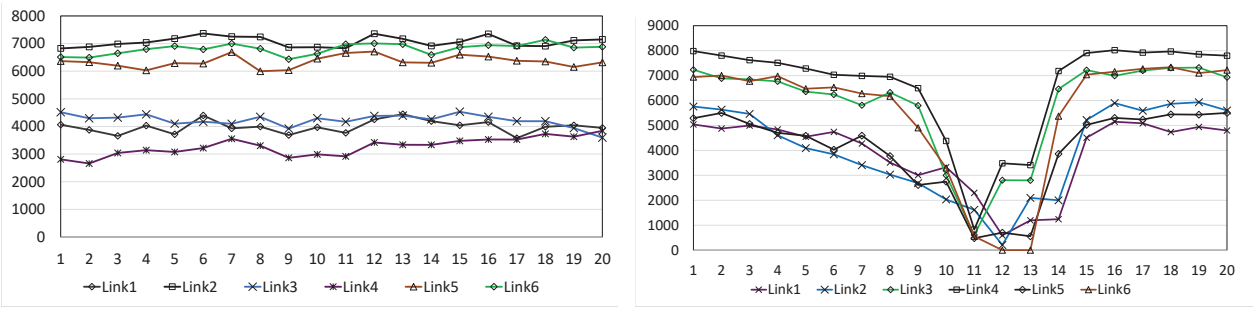
Next we explain the power model for the proposed hybrid network. The power of our proposed hybrid NoC can be partitioned into two main contributions: wired network and wireless network.

$$E_{total} = E_{wired} + E_{wireless} \quad (1)$$

$$E_{wireless} = E^{dynamic}_{wireless} + E^{static}_{wireless} \quad (2)$$

$$E^{static}_{wireless} = (P^{tx}_{static} + P^{rx}_{static}) \times C \times T \quad (3)$$

$$E^{dynamic}_{wireless} = (P^{tx}_{dynamic} + P^{rx}_{dynamic}) \times b/R \quad (4)$$



(a) MUM (b) StreamCluster

Fig. 5: Wireless link usage for benchmark MUM and StreamCluster. Three most and least used links are measured in 20 epochs. X-axis represents epochs and Y-axis represents link usage counts.

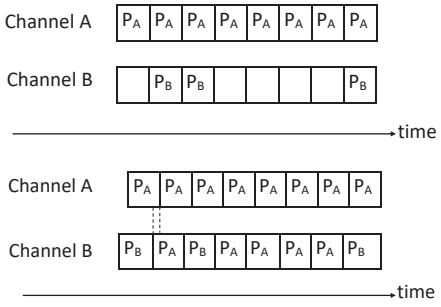


Fig. 6: Adaptive Wireless Channel Sharing.

The hybrid network power consumption consists of both wired and wireless contributions. We use the built-in power model of GPGPU-Sim to evaluate wired network. Here we explain the wireless power model which is described in the Equation (1)-(4). The model is similar to the one developed in [20] where C and T in equation 3 are the total number of execution cycles and system clock period. In equation 4, b and R are total bits transmitted and wireless channel data rate. P^{tx}_{static} and P^{rx}_{static} are static values of transmitters and receivers. Dynamic transceiver power is denoted by $P^{tx}_{dynamic}$ and $P^{rx}_{dynamic}$ respectively. We retrieve these values using NoXim [17] and use GPGPU-Sim to collect other statistics such as simulation cycles.

B. Adaptive Wireless Bandwidth Allocation

One of the key advantages of wireless networks over point-to-point networks (such as metal or nano-photonics) is their reconfigurability. A single channel can be shared between multiple users, greatly reducing wiring complexity, area overhead and design cost. Routers can also make changes to their channels at run time, by adjusting the transmitting frequency. To efficiently utilize the available bandwidth, we propose a novel scheme that adaptively allocate wireless bandwidth to channels based on usage requirement.

We first analyzed the wireless link usage in our proposed network using benchmark *MUM* and *StreamCluster*. In Figure 5, we plot usage of the three most heavily used and three most lightly used links in 20 epochs. It can be observed that for *MUM*, the ratio between the highest to lowest link usage is in the range of 2.3-1.5 times. For *StreamCluster*, the ratio is about 2.8 to 1.4 times. Such imbalance leads to wasted bandwidth because some links are allocated more resource than they need while busy links do not have enough bandwidth.

Although there have been schemes about bandwidth allocation in wireless communication, we cannot simply borrow them into the on-chip networks. For example, we can proportionally divide total bandwidth among all links based on their usage and allocate the new bandwidth in next epoch. This method may lead to optimal bandwidth usage, but it is not suitable for the on-chip environment because it incurs huge cost and design complexity.

We employ a light weight scheme called "Borrow from the Rich". Instead of reallocating bandwidth to all links, our scheme only involves part of the links. The idea is to assign all links with equal bandwidth and allow a busy link to share bandwidth with a less used link. At the end of each epoch, we pick links with the highest usage and put them into a "borrowing pool". Similarly, we create a "lending pool" using least used links. Then we pick a link from each pool and build a token ring between them to share bandwidth of the lending link. The borrowing link now has its bandwith increased, including its original bandwidth and part of the lending link's bandwidth.

Figure 6 illustrates how this scheme works. Initially link A and link B are assigned same amount of wireless bandwidth. However, link A is congested but link B is idle half of the time. To fully utilize link B's bandwidth, we allow link A and B to share the same wireless bandwidth that is originally allocated to link B. To avoid conflict, a token ring is formed between link A and B. At the same time, link A still has its private channel. So whenever link B has no packet to send, it sends a token to link A's router, then link A can send a packet using their shared bandwidth. Note there is a small amount of delay overhead that is caused by token passing in our scheme.

IV. Evaluation

A. Methodology

We use GPGPU-Sim [1] to simulate our proposed hybrid network. Table 1 shows the configuration used in our evaluation. Our baseline network consists of a 8x8 2D mesh with 56 computing cores and 8 memory controllers. The overlaid wireless network is implemented as 2x4 mesh. We choose mesh topology for wireless network because it has low design

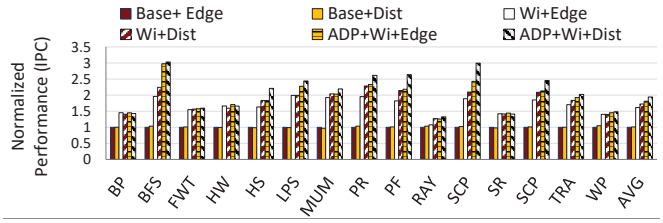


Fig. 7: Normalized performance in a 64-node NoC.

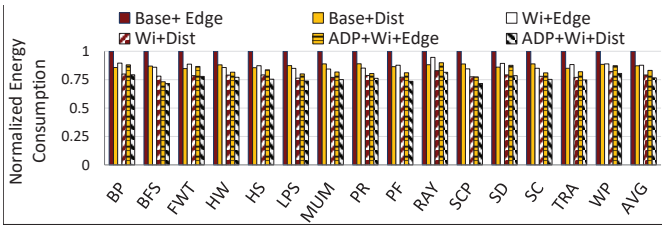


Fig. 8: Normalized energy consumption in a 64-node NoC.

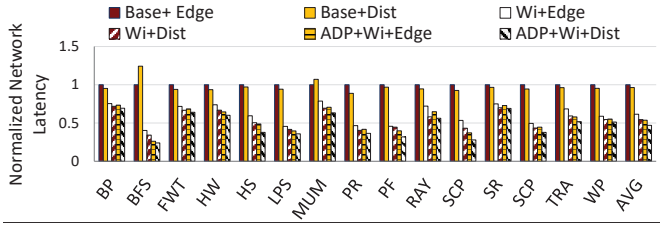


Fig. 9: Normalized network latency in a 64-node NoC.

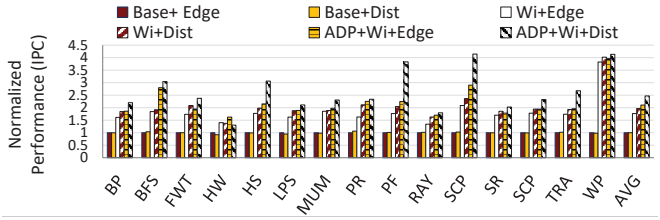


Fig. 10: Normalized performance in a 256-node NoC.

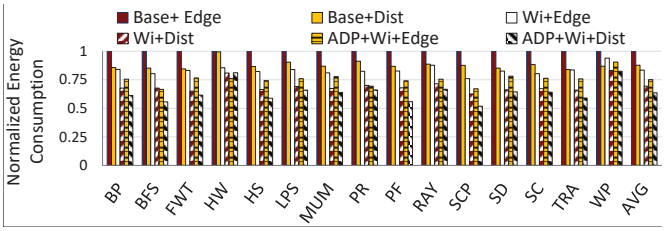


Fig. 11: Normalized energy consumption in a 256-node Noc.

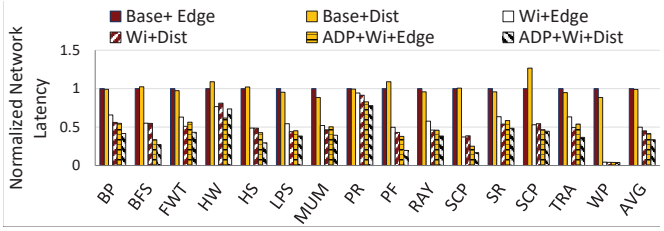


Fig. 12: Normalized network latency in a 256-node NoC.

complexity and simple routing algorithms. We experimented with two types of MC placements (edge and distributed) in order to observe their impacts on the hybrid wireless network. We used GPU workloads from ISPASS [1], Rodinia [2] and Cuda SDK [3] to evaluate our design. To simulate the power consumption, we collect parameters using NoXim [17]. The wireless link delay is determined by flit size, channel data rate and system clock frequency. We set flit size to 128 bits and wireless channel data rate to be 20Gbps, so it takes 6.4 ns to transfer one flit. We simulate a GPU system similar to NVIDIA GTX480 which has a clock frequency of 1.4 GHz. Therefore the single hop wireless delay is 9 clock cycles.

Shader Core	56 cores, 1.4GHz, SIMT width=8
Warp Scheduler	Greedy-Then-Oldest
Shared Memory	48 KB
Cache	2KB L1 I-Cache (4 sets/4 ways LRU), 16KB L1 D-Cache (32 sets/4 ways LRU), 64KB L2 Cache per MC (8 way LRU)
Memory Model	8 MCs, 924 MHz
Wired NoC	128-bit channel width, 2-stage pipeline, 16-byte flits, 1-cycle link latency, X-Y routing, vc buffer depth=4
Wireless NoC	128 bit flits, 4 flits per packet, X-Y routing, 9-cycle link latency, transmit energy=0.5 pJ/bit, receive energy=0.7 pJ/bit, epoch interval=50000
subnet	2

TABLE I: System configuration.

B. Performance and Power Analysis

The performance evaluation of our proposed hybrid wireless network architectures is shown in Figure 7. We compare six sets of results with different MC placements and both static and adaptive wireless bandwidth allocation schemes. The results are normalized to a baseline 8x8 mesh network. In most cases, Distributed MC placement has better performance than Edge placement. The hybrid architecture greatly improved the network performance. On average, the hybrid wireless network improved performance by 61% for Edge MC placement and 72% for Distributed MC placement. The adaptive bandwidth allocation scheme further improves performance for both MC placements by about 81%and 94%. *BFS* and *SCP* achieve the most performance gain and their IPC are increased by 3 times. All benchmarks achieve more than 1.5 times performance gain except *BP*.

Figure 8 shows the energy consumption of the six NoC architectures under evaluation. Energy consumption of all NoCs are normalized to base line with Edge MC placement. It can be observed that Distributed MC placement consumes less energy compared with Edge placement. This is because MCs are located near the center of the network and reduced average transmission distance. Comparing with baseline, static bandwidth allocation scheme can save energy by 13% and 21% averagely for the two MC placements. With adaptive bandwidth allocation, energy savings are improved by 17% and 24% on average. This is because wireless links have very low power consumption compared with metal wires and the global network significantly reduced hop counts in the network. We also evaluated network latency as show in

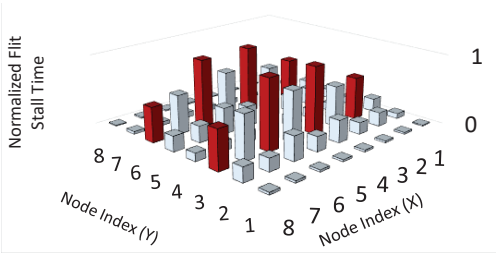


Fig. 13: Flit stall time in routers of hybrid network for workload WP.

Figure 9. The hybrid NoC reduce network latency by 50% on average. *BFS* and *SCP* receive maximum decrease in network latency which translates to largest performance gain.

To evaluate the impact of our hybrid NoC on network scalability, we performed sensitivity analysis using a network with 256 nodes. Figure 10, Figure 11 and Figure 12 show the evaluation results for performance, energy and network latency respectively. Our results show the GPU NoCs receive even more benefit from our hybrid design as network size increases. On average, our proposed NoC can improve IPC by 2.47 times and reduce energy consumption by 37%. The hybrid wireless NoC exhibits good scalability and provides a promising solution in designing large GPU systems.

Figure 13 shows the flit stall time for workload *WP* in our hybrid NoC. The flit block time is normalized to the longest one, so the absolute value does not represent real time. Instead the important information is the relative flit block time in all routers. When we compare Figure 13 with Figure 2(b), we can find the difference between MCs and other routers are decreased. This means traffic is more evenly distributed among the routers and our design can effectively alleviate congestion in traffic bottlenecks.

Next we discuss the overhead of our hybrid NoCs. It is estimated that wireless transceiver and digital part of a wireless router is around 0.3 mm^2 respectively [12]. The overall area of a wireless router is estimated to be 0.8 mm^2. If the network consists 8 wireless routers, the overall area overhead is 5.6 mm^2. Comparing with the size of NVDIA GTX480 GPU which is 529 mm^2, the area overhead of wireless routers is negligible.

V. Related Work

To reduce cost, a checkerboard architecture was proposed to design GPGPU NoCS [6]. Through VC monopolization and employing asymmetric request and reply networks, Jang et. al. proposed a bandwidth efficient NoC design [4]. Kim et.al proposed a conflict-free design for the reply network called DA2mesh [10] which assigns each memory node a dedicated channel-sliced network. There are other GPGPU NoC schemes such as asymmetric cmesh [9] and ring-chain network [8]. However, these scheme treats MC connected routers same as other router when allocating network bandwidth and hardware resource, wasting resource on non-congesting routers. Compared with those schemes, our design improves network performance by removing network bottleneck and enhancing NoC scalability.

VI. Conclusion

In this paper, we analyzed the network bottleneck of on-chip communication within GPUs. We propose hybrid NoC architectures to design the reply network and employ wireless links to remove network bottlenecks. We also provided solutions to critical challenges in designing such networks. Our experiment results show that the proposed design can not only achieve better power-performance efficiency but also improve network scalability.

References

[1] A. Bakhoda, G. Yuan, W. Fung, H. Wong, and T. Aamodt. Analyzing CUDA workloads using a detailed GPU simulator. In *ISPASS*, 2009.
[2] S. Che, M. Boyer, J. Meng, D. Tarjan, J. Sheaffer, S.-H. Lee, and K. Skadron. Rodinia: A Benchmark Suite for Heterogeneous Computing. In *IISWC*, 2009.
[3] NVIDIA. CUDA C/C++ SDK Code Samples. 2011. [Online]. Available: http://developer.nvidia.com/cuda-cc-sdk-code-samples
[4] H. Jang, J. Kim, P. Gratz, K. Yum, E. Kim. Bandwidth-Efficient On-Chip Interconnection Designs for GPGPUs. In *DAC*, 2015.
[5] T. Krishna, A. Kumar, P. Chiang, M. Erez, Li-S. Peh. NoC with near-ideal express virtual channels using global-line communication. In *HOTI*, 2008.
[6] A. Bakhoda, J. Kim, T.M. Aamodt. Throughput-Effective On-Chip Networks for Manycore Accecerators. In *Proc. MICRO*, 2010.
[7] X. Zhao, S. Ma, Y. Liu, L. Eeckhout, Z. Wang. A low-cost conflict-free NoC for GPGPUs. In *Proc. DAC*, 2016: 34:1-34:6.
[8] X. Zhao, S. Ma, C. Li, L. Eeckhout, Z. Wang. A heterogeneous low-cost and low-latency Ring-Chain network for GPGPUs. In *Proc. ICCD*, 2016: 472-479.
[9] A. Kavyan, J. L. Abellan, Y. Ma, A. Joshi, D. Kaeli. Asymmetric NoC Architectures for GPU Systems. In *Proc. NoCs*, 2015.
[10] H. Kim, J. Kim, Wong. Seo, Y. Cho, S. Ryu. Providing Cost-effective On-Chip Network Bandwidth in GPGPUs. In *Proc. ICCD*, 2012.
[11] Y. Pan, P. Kumar, J. Kim, G. Memik, Y. Zhang, A. Choudhary. Firefly: Illuminating Future Network-on-Chip with Nanophotonics. In *ISCA*, 2009.
[12] A. Ganguly, K. Chang, S. Deb, P. P. Pande, B. Belzer, C. Teuscher. Scalable Hybrid Wireless Network-on-Chip Architectures for Multicore Systems. In *IEEE Transactions on Computers*, 2011.
[13] S. Lee, et.al. A Scalable Micro Wireless Interconnect Structure for CMPs. In *MobiCom*, 2009.
[14] S. Deb, A. Ganguly, P. P. Pande, B. Belzer, D. Heo. Wireless NoC as Interconnection Backbone for Multicore Chips:Promises and Challenges. In *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2012.
[15] D. DiTomaso, A. Kodi, D. Matolak, S. Kaya, S. Laha, W. Rayess. Energy-efficient adaptive wireless NoCs architecture. In *NoCs*, 2013.
[16] H. Gu, J. Xu, W. Zhang. A low-power fat tree-based optical network-on-chip for multiprocessor system-on-chip. In *DATE*, 2009.
[17] V. Catania, A. Mineo, S. Monteleone, M. Palesi, D. Patti. Noxim: An open, extensible and cycle-accurate network on chip simulator. In *ASAP*, 2015.
[18] X. Yu, J. Baylon, P. Wettin, D. Heo, P. Pande, S. Mirabasi. Architecture and Design of Multi-Channel Millimeter-Wave Wireless Network-on-Chip. In *IEEE Design and Test*, 31(6):19-28, 2014.
[19] N. Mansoor, A. Ganguly. Reconfigurable Wireless Network-on-Chip with Dynamic Medium Access Mechanism. In *NoCs*, 2015.
[20] S. Abadal .et al. On the area and energy scalability of wireless network-on-chip: A model-based benchmarked design space exploration. In *IEEE/ACM Transactions on Networking*, 2015.
[21] S. Gade, S. Deb. HyWin: Hybrid Wireless NoC with Sandboxed Sub-Networks for CPU/GPU Architectures. In *IEEE Transactions on Computers*, VOL. 66, NO. 7, JULY 2017.
[22] W. Choi, K. Duraisamy, R. Kim, J. Doppa, P. Pande, R. Marculescu, D. Marculescu. Hybrid network-on-chip architectures for accelerating deep learning kernels on heterogeneous manycore platforms. In *CASES*, 2016.