A Low-cost Conflict-free NoC architecture for Heterogeneous Multicore Systems

Yuwen Cui*, Hui Zhao*, Shakthi Prabhakar*, Saraju Mohanty* and Juan Fang†
*Department of Computer Science and Engineering, University of North Texas
†Faculty of Information Technology, Beijing University of Technology
Email: cuiyuwen@my.unt.edu, hui.zhao@unt.edu, shakthiprabhakar@my.unt.edu,
saraju.mohanty@unt.edu, fangjuan@bjut.edu.cn

***Abstract*—Heterogeneous multi-core systems integrate general-purpose CPUs and data-parallel GPUs on a single chip. However, the integration between CPUs and GPUs causes severely interference both on CPU request and GPU request. Because CPUs and GPUs have diverse sensitivity to network performance regarding latency and throughput, there exists severe interference between their data communication when they share the same Network-on-Chip (NoC).**

In this paper, we propose an interference-free NoC architecture to meet this demand. Specifically, our proposed scheme reduces the network interference effectively through MCs partition, specially designed routing algorithm, and bypass scheme for interference mitigation in heterogeneous systems. By evaluating different CPU and GPU applications, we found that our proposed NoC architecture is able to improve the overall network performance as well as the overall system performance. Simulation results show that the proposed scheme can reduce over 17% of energy consumption on the average compared with baseline heterogeneous multi-core architecture. Also, the average performance of CPU can be improved as much as 30% and GPU average performance can be increased over 9%, compared to the baseline 6x6 mesh network.

***Index Terms*—Heterogeneous multicores, Network-on-Chip (NoC), conflict-free NoC**

I. Introduction

Heterogeneous multi-core systems have been applied to various computing platforms such as high-performance servers, personal computers, handheld devices and gaming consoles. Representative products include AMD's Fusion APUs [13], Intel's Sandy Bridge [9] and ARM's MALI [4] that integrate general purpose programmable GPUs together with CPUs on a same die. Such state-of-art designs enable faster communication by allowing CPUs and GPUs to share the same memory and some designs even provide a unified virtual address space and programming model for both CPU and GPU applications.

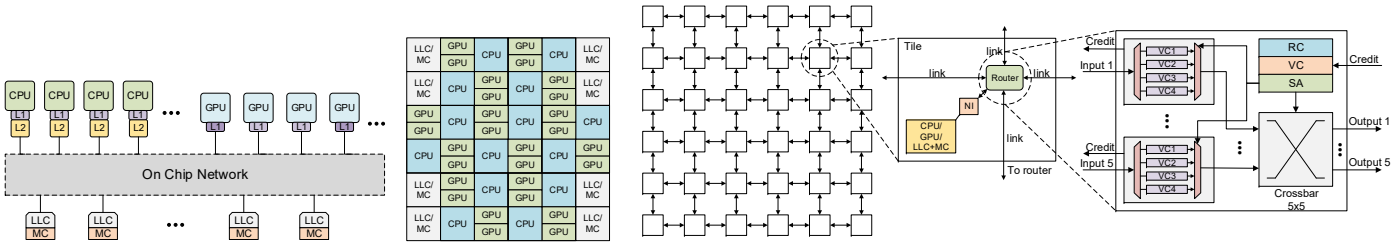
To be more specific, sharing memory between CPUs and GPUs of these integrated heterogeneous architectures exhibits several advantages. Firstly, such designs can improve the performance by reducing communication overhead because no explicit data transfer is needed between CPUs and GPUs. Secondly, the fused multi-core architectures can reduce energy and resource costs due to better resource utilization. Thirdly, programming models for such systems become simpler because no explicit GPU memory management is needed. The reduced communication costs and increased bandwidth have the potential to enable new optimizations that were previously hard to achieve. As a result, this design paradigm enables new opportunities that can be exploited to enhance performance and reduce system cost.

In heterogeneous multicores, the interference between CPU and GPU traffic can lead to severe performance degradation [6], [20], [21]. This is because CPUs and GPUs exhibit very different traffic patterns: CPU cores generate moderate coherence traffic and is very sensitive to latency, while GPUs generate a large amount of streaming traffic and require high network throughput to satisfy the demand from their data-parallel processing. Without careful design, these two types of traffic will contend for the shared network resource such as buffers, switches and link bandwidth. Therefore, it is critical for NoC designs need to efficiently manage the resource sharing between CPUs and GPUs in order to achieve optimal performance.

An intuitive approach to resolve the traffic interference is to enforce isolation through multiple networks. CPUs and GPUs can inject their traffic into separate networks and this will totally remove interference. However, this technique has two disadvantages: firstly, static partitioning of the network resource may fail to satisfy dynamic demand from various applications; secondly, multiple networks significantly increase the cost of NoCs. Under the stringent budget of area and energy in a heterogeneous system, optimal solutions need to be developed in order to avoid interference while maintaining low cost. Although NoC designs have matured in CPU-based and GPU-based multi-cores [5], [11], [14], the design of interference-free NoCs for heterogeneous systems is largely unexplored. Only a handful of works have examined the impact of NoC design in heterogeneous systems [1], [19], [20]. These work focus on improving network performance but ignored the cost which is a most important constraining factor of the heterogeneous system design. Thus, it is of primary importance to develop NoC designs with both interference avoidance and cost-efficiency.

In this work, we observe that the interference between CPU/GPU traffic can be avoided by properly designed routing algorithms, obviating the need of physically partitioned networks. We propose a CPU and GPU traffic interference free NoCs scheme that using one shared physical network.

Our proposed technique avoids interference in three ways:



(a) Heterogeneous CPU-GPU architecture. (b) 2D mesh layout. (c) NoC Router Micro Architecture.

Fig. 1. Heterogeneous multi-core and NoC architecture.

- We isolate CPU and GPU conflicts through partitioned memory controllers (MCs) in terms of CPUs and GPUs do not share a same resource in the same MC.
- We use XY-YX routing to separate the CPU and GPU flow both on request and reply traffic
- We employ repeater to each router where do not need to route the CPU or GPU traffic.

II. Background and Motivation

Figure 1(a) illustrates the high level view of our baseline heterogeneous CPU-GPU multi-core architecture. Throughput oriented GPU cores and latency oriented CPU cores are connected with shared LLC and memory controllers (MC) by a NoC. In order to increase the system scalability, the CPU and GPU cores are organized in a tiled structure similar to prior work [12], [21]. Figure 1(b) shows the layout of our baseline architecture connected with a 6×6 mesh network. The CPU and GPU cores, LLC and MCs are attached to the routers of the NoC.

In this layout, there are 14 CPUs, 28 GPUs and 8 memory slices in total. We organize CPU and GPU cores into 7 processing tiles with each tile consisting of 4 GPU cores and 2 CPU cores. The reason why we choose a 2:1 ratio for GPUs to CPUs is because a single GPU core (i.e. streaming multiprocessor or SM) in Fermi GF110 (45nm technology) occupies roughly half the area of an Intel Nehalem CPU core (45nm technology). Figure 1(c) depicts the microarchitecture of a generic NoC router used in our baseline mesh network. Each router has five ports with one port connecting with one of the four direct neighbors and one port connecting with local Processing Element (PE). In this case, the PE can be a CPU core, a pair of GPU cores and a memory slice (containing a LLC and a MC). We use virtual channels to organize input buffers of each router for better flow control. Each input port has four virtual channels (VCs). The arriving flits are first stored in a VC before going through the router pipeline stages.

Figure 2 shows the performance of CPU applications between running alone and running together with a GPU application *MUM*. Performance degradation can be observed for all CPU applications, with an average of 41%. Many of the CPU applications can only achieve half of their stand-alone IPC. We also characterize the GPU performance degradation incurred from CPU interference as shown in Figure 3. In this case, we fix the multiprogrammed CPU applications and run a different GPU application each time. On the average, GPU suffers 14%

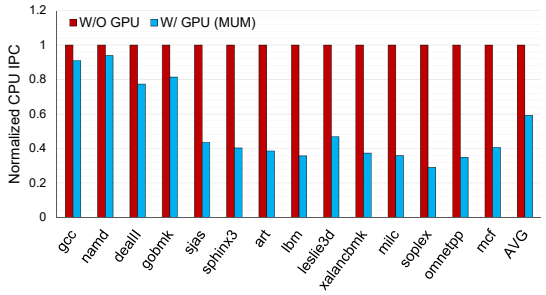


Fig. 2. The performance of CPU applications running with and without GPU applications.

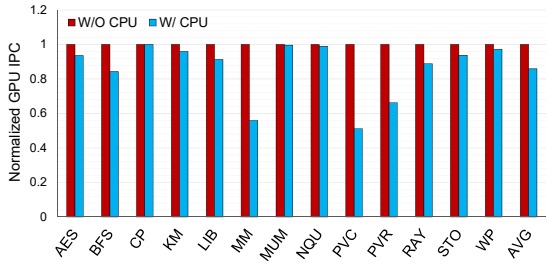
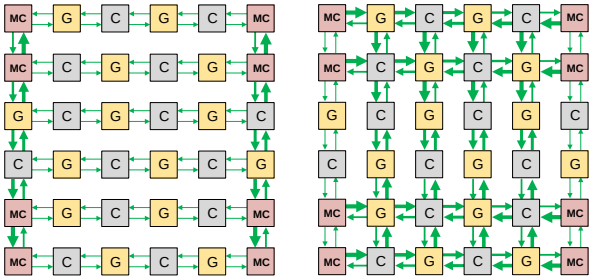


Fig. 3. The performance of difference GPU applications running with and without CPU applications.

degradation and the performance degradation for most of the GPU applications is around 10%. Applications such as *MM, PVC* and *PVR* receive the most significant performance loss due to CPU interference.

Our characterization shows that there exists severe interference between CPUs and GPUs in the shared network and mitigation schemes must be developed in order to reduce performance degradation. An intuitive solution is to enforce network isolation using separate networks. Multiple physical networks have been proposed for separating throughput sensitive and latency sensitive CPU applications [2] and for energy reduction [15]. However, such techniques incur significant hardware overhead. Considering the overhead of NoCs already accounts for 30% of the overall chip cost [17], simply splitting the interfering traffic into separate networks is not a cost efficient solution. In addition, separate physical networks may not achieve optimal resource utilization since resource from one network during its idle or low active period cannot be utilized by another network. This motivated us to develop the NoC architecture proposed in this work to enhance performance for both CPUs and GPUs through interference mitigation.



(a) XY Request (b) XY Reply

(c) X-Y Request (d) Y-X Reply

Fig. 4. Network traffic for XY routing and XY-YX routing.

III. Interference-free NoC Architectures

In this section, we will describe the details about the proposed NoC architectures for interference mitigation NoCs in heterogeneous multi-core systems.

Compared with the baseline mesh NoC architecture, the proposed NoC architecture can effectively reduce the traffic interference. We first observe the interference of shared MCs between CPU and GPU traffic in order to find an efficient scheme to reduce communication conflicts from a core to a memory slice. Then, we employ private MCs between CPU and GPU with specific MCs placement to avoid CPUs and GPUs' traffic interference. According the specific placement of MCs, we develop a routing algorithm specially tailored to avoid traffic interference. The advantage of the proposed routing algorithm is that it separates traffic in all routers into different dimensions so that no contention between CPUs and GPUs will occur in these routers. The routing algorithm can also reduce more than 40% links compared with the baseline mesh. Then we propose to employ repeater to bypass CPU flits from the router which is connected to GPU and bypass the GPU flits from the router which is connected to CPU core.

Our proposed interference-free NoC architecture uses a 6x6 mesh topology with 8 MCs: 4 MCs are used for CPUs and 4 MCs are used for GPUs. In order to keep the tile-based architecture, we keep the same MCs placement compared with baseline architecture. But, the difference of MCs placement compared with baseline architecture is that we place the left side 4 MCs as the CPUs' shared MC and the right side 4 MCs as the GPUs' shared MC. With our MCs' placement scheme, we remove interference from the link between CPUs and GPUs.

GPU and CPU applications have different requirements from the networks: GPUs are bandwidth-sensitive and CPUs are latency-sensitive. It is difficult to satisfy both CPU's and GPU's requirements inside a same router. A routing algorithm is a critical design factor affecting the performance of a NoC and directly determines the amount of traffic each link will carry. To find a routing algorithm that can mitigate interference between CPUs and GPUs, we first analyzed the impact of different dimension routing algorithms as shown in Figure 4. In this paper, we propose to use XY-YX routing algorithm to separate CPU and GPU traffic for internal routers. Figure 4(a) shows the request (core-to-MC) traffic pattern under XY routing with shared MCs. Figure 4(b) shows the reply (MC-to-core) traffic pattern under XY routing with shared MCs.

Algorithm 1 XY-YX Routing Algorithm

```
// f_sta is flit of each packet, sta is the status of each flit,
sta = 0 means the GPU request flits, sta = 1 means the
GPU reply flits, sta = 2 means the CPU request flits, sta =
3 means the CPU reply flits.
// C is current node of f_sta, D is destination node of f_sta.
if sta = 0 or sta = 2 then
    if C and D in different column then
        Choose straight route to East or West;
    else
        //C and D in same column
        Choose straight route to North or South;
    end if
else if sta = 1 or sta = 3 then
    if C and D in different row then
        Choose straight route to North or South;
    else
        //C and D in same row
        Choose straight route to East or West;
    end if
end if
```

As can be observed from Figure 4, the reply network has much heavier traffic loads than the request network using XY routing. In our proposed scheme, we apply XY-YX routing algorithm to cluster-based NoC architecture. We employ XY routing for CPU request and GPU reply traffics, and YX routing for CPU reply and GPU request traffics. The details of the proposed XY-YX routing algorithm is shown in Algorithm 1. Under our routing algorithm, CPU and GPU traffic will not interfere with each other in the same dimension. This is because CPU and GPU packets will not go through a same dimension in these routers. Network resources such as vc buffers, crossbar switches, output ports and links will not be shared by both CPU and GPU packets and thus no interference will occur. The traffic loads of XY-YX routing algorithm for NoC-based heterogeneous multi-core system is depicted in Figure 4(c) and Figure 4(d). In Figure 4(c) and Figure 4(d), the green arrow and red arrow represent CPU and GPU traffic respectively. Compared with baseline XY routing algorithm, applying XY-YX routing algorithm to our NoC architecture can not only reduce the interference between CPUs and GPUs but also can reduce the utilization of links. On the other

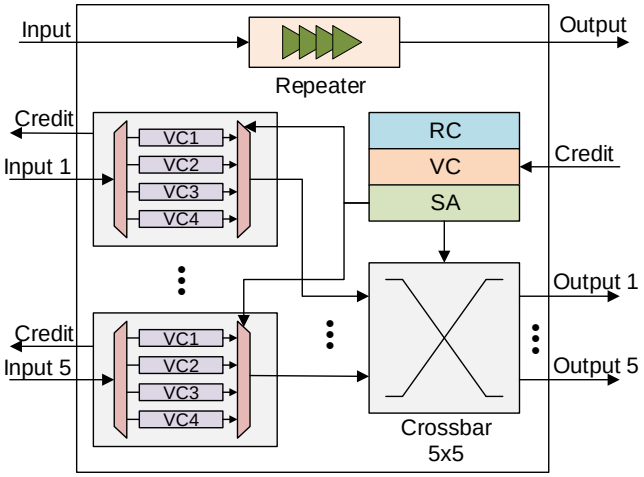


Fig. 5. Our proposed router architecture incorporated with repeater.

hand, network congestion is also reduced compared with conventional XY routing algorithms. Therefore, the proposed routing algorithm will result in improved network performance since the interference is removed.

Deadlock Avoidance: In the baseline NoC, XY routing algorithm can avoid deadlocks by avoiding turns (Y-to-X) [8]. However, our proposed scheme employs X-to-Y and Y-to-X to operate in the same network. We need to ensure that deadlock will not occur in the proposed design. According to Figure 4(c) and Figure 4(d), the CPU and GPU request packets first traverse along the *x* dimension and then along the *y* dimension, the CPU and GPU reply packets traverse in the opposite way. Different from CPUs, GPUs request packets traverse along the *x* dimension first and then along the *y* dimension, and the CPUs, GPUs reply packets go through an opposite way. On the other hand, the request and reply network is physical isolation. This means that the request traffic and reply traffic use different networks to reach their destination. Protocol deadlock will be activated if we only use on one physical network to transfer request and reply packets.

Through the prior description about XY-YX routing algorithm, the interference between CPU and GPU can be removed for heterogeneous multi-core systems. Furthermore, the vertical links are not used for the non-edge routers according to Figure 4(c) and Figure 4(d). Also, we can remove the link between the routers of CPU and GPU based on our NoC architecture. So, we apply non-buffer repeater to our NoC architecture.

As show in Figure 5, we add non-buffer repeater to our router architecture which connects the CPU and CPU cores or GPU and GPU cores directly. This technology can directly connect two routers which are not closed. That means the router of CPU cores connect without the router of GPU cores, and vice versa. So, the CPU performance will acquire lots of benefits due to the latency reduction. For example, if a GPU core places between in two CPU cores, the non-buffer repeater of this GPU router can bypass CPU flits directly rather than traversing the router again. For a large scale NoC architecture of heterogeneous multi-core systems, this will provide a graceful performance degradation. On the other hand, applying the non-buffer repeater to our NoC architecture not only improves the CPU performance but also reduces the energy consumption. Because both the CPU and GPU flits decrease a multitude of unnecessary hops for a network communication. Therefor, it can reduce the energy consumption for routing and switching.

IV. Evaluation Methodology

In this section, we will demonstrate our experimental setup and the benchmarks for evaluation.

System Setup. To evaluate our proposed schemes, we integrate GPGPU-Sim v3.x [3] with an in-house cycle-level x86 CMP simulator. Each simulation warms up with 500K instructions before executing GPU and CPU instructions. To measure CPU performance, we run until the slowest CPU core reaches 5 million instructions. To measure GPU performance, we run the applications until completion or 100 mullion instructions, whichever comes first.

Table I shows the configuration details of GPU and CPU cores. The baseline NoC architecture uses 28 GPU cores and 14 CPU cores. Each GPU core contains 32-wide SIMD lanes and is equipped with an instruction cache, private L1 data cache, constant, and texture caches. Each CPU core is a 3-way issue x86 core with private write-back L1 instruction/data cache and a L2 cache.

TABLE I
Baseline Heterogeneous CPU-GPU architecture configuration

GPU core config.	28 shader cores, 1400MHz, SIMT Width = 16 × 2
GPU resources/core	Max. 48 warps/core, 32 threads/warp, 7 48KB Shared Memory, 32684 Registers
GPU caches/core	16KB 4-way L1 data cache, 12KB 24-way texture cache, 8KB 2way constant cache, 2KB 4-way I-cache, 128B line size
CPU core	16 x86 cores, 2000 MHz, 128-entry instruction window, OoO fetch and execution
CPU L1 cache	32KB 4-way, 2 cycle lookup, 128B line size
CPU L2 cache	256KB 8-way, 8 cycle lookup, 128B line size
Share SRAM LLC	1 × 8 MB, 128B line, 16-way
Interconnect	6 × 6 shared 2D mesh, 14000MHz, XY-YX routing, 2 GPU cores per node, 1 CPU core per node, 32B Channel Width, 4VCs, Buffers/VC = 4
Memory Model	8 Shared GDDR5 MCs, 800 MHz, FR-RCFS, 8 DRAM-banks/MC

Workloads and Applications. We select several GPU and CPU applications to evaluate our proposed scheme. Our simulation uses 16 GPU applications from ISPASS2009 [3], Mars [10], Parboil [16], Rodinia [7]. For CPUs, we run multi-programmed workloads with a mix of applications from different application suites including scientific, commercial, and desktop applications drawn from the SPEC CPU 2000/2006 INT and FP suites and commercial server workloads. Furthermore, we conduct workload analysis and select 14 CPU benchmarks that represent a wide range of MPKI values (Miss-

per-kilo-instructions). The selected CPU benchmarks are listed in Table II.

TABLE II
CPU APPLICATIONS

CPU app. category	Applications	L2 MPKI range
Low	povray, namd dealII, gobmk	[0.2, 2.3]
Medium	sjas, astar, sjbb, ocean, libquantum, lbm	[4.8, 22]
high	milc, soplex, omnetpp, mcf	[25, 112.4]

Performance Metrics. To measure GPU performance, we use GPU speedup (SU_{GPU}), which is the ratio of its instruction-per-cycle (IPC) when it runs along with CPU to its IPC when it runs alone. We use weighted speedup to capture CPU performance (WS_{CPU}). WS_{CPU} is defined as $\sum_{i=1}^{n}(IPC_{i,multiprogram/IPC_{i,alone}})$, where n is the number of CPU applications in the workload. All average speedup results in this paper use harmonic mean. Based on user preferences, one might want to change the importance given to CPU or GPU when calculating speedup.

Energy Metrics. To evaluate the energy consumption of our proposed scheme, we measure the maximum electrical and optical power consumption. For the electrical components, we use Booksim [18] to evaluate power consumption according to the chip area, NoC size, VC numbers and etc. All power parameters of electrical components are extracted from the configure file provided by Booksim.

V. RESULTS AND ANALYSIS

In this section, we analyze and evaluate our proposed interference-free NoC architecture with several schemes. The baseline is a mesh network without traffic partition. To compare with our interference-free schemes, we also evaluate *Physical Network Partition* which equally divides request and reply network between CPU and GPU applications for physical partitioning the network. We also experimented with a scheme of private MCs between CPU and GPU based on baseline. Interference-free NoC is our proposed interference mitigation scheme employing private MCs, routing and flits bypass technology.

A. System Performance

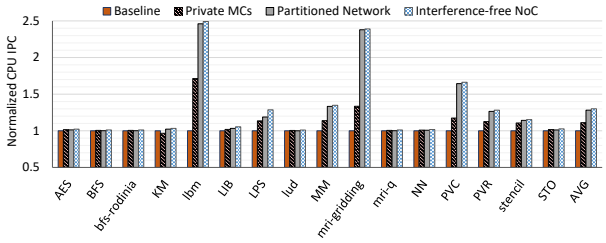
We first evaluate and analyze the impact of interference mitigation of proposed techniques on system performance. Figure 6(a) and Figure 6(b) depict the results of CPU and GPU performance, which are normalized to the baseline network performance. As shown in 6(a), the first bar shows the performance of the baseline network and the second bar is the performance of physical MCs partition between CPU and GPU applications. We also explore the influence of system performance for physical network partitioned and the results are shown as bar 3. The fourth bar is the performance of the our proposed interference-free technique which combines XY-YX routing algorithm and bypass technology together. As can be observed in Figure 6(a), the average CPU IPC of our interference-free scheme increases performance more than 30% compared with the baseline. This is because the our interference-free scheme can avoid GPUs to take away resources from CPUs when GPU traffic load is high both in the link and MCs. However, it cannot allow CPUs to utilize idle resource when GPU is not sending a lot of traffic as did in the baseline round-robin allocation techniques. So the static equal partition cannot improve performance because the resource is not optimally allocated to the users who really need it. The results also show that the overall CPU performance is also significantly improved when the NoC architecture uses physical partitioned network. This is because the CPUs and GPUs request or reply traffic is totally interference free in terms of then use their request or reply network separately. But this will lead to a significant energy consumption due to the doubling of the physical network channels. For our interference-free technique, we can observe that the CPU IPC is enhanced by over 250%. Some applications achieve significant improvement such as 14 CPU applications co-run with *lbm, mri-gridding and PVC*. This due to two main reasons: 1) the interference-free scheme not only removes interference between the CPUs and GPUs in traffic channels but also reduces the traffic congestion by separate MCs to private MC between CPU and GPU so that the traffic load both in the wired network and MC is greatly reduced; 2) the bypass technique can reduce the hops on a long range with low latency so that the overall network latency is significantly reduced.

As can be observed in Figure 6(b), the GPU performance improved by 9% on average with our interference-free scheme. *lbm* suffers from the most significant performance degradation among all applications. This is because this application has very intensive traffic and our schemes effectively constrained the resource taken by the GPUs so that the CPU performance will not degrade significantly. As can be observed, the interference-free techniques achieve more or less performance improvement among all schemes compared with physical network partitioning method. It can also be observed in Figure 6(a) and Figure 6(b), interference-free scheme helps to improve performance in several applications. Because our interference-free scheme not only reduces the latency of CPU packets but also mitigates the traffic congestion for GPU packets.

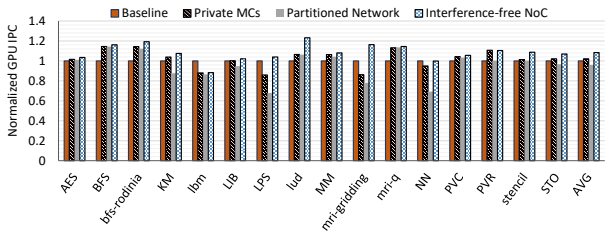
B. Energy Consumption

In our design, we utilize repeater in the router which is connected to CPUs and GPUs core to connect the CPUs or GPUs separately.

Figure 7 shows the overall energy consumption of our proposed techniques. Our schemes can reduce the average energy consumption around 17% than the baseline NoC architecture. As can be observed, the average power consumption of physical partitioned network is increased by 55% compared with baseline architecture due to the doubling of the physical channels. For our proposed NoC architecture, the optical links can reduce a lot of power consumption, and our schemes can also reduce over 40% buffers and links for the baseline



(a) The overall CPU performance with different interference mitigation scheme.



(b) The overall GPU performance with different interference mitigation scheme.

Fig. 6. The overall system performance.

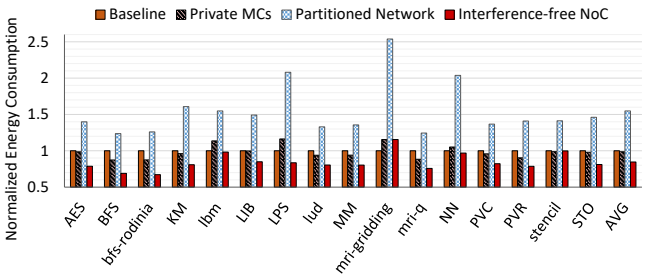


Fig. 7. The overall energy consumption.

mesh network. Compared with other GPU applications, the energy consumption of our proposed NoC architecture is the lowest one than other techniques. This is because proposed NoC schemes not only removes the vertical links by routing algorithm of non-edge routers but also reduces the power consumption of router by bypass technique.

VI. Conclusion

In this paper, we explore, design and evaluate an interference-free NoC architecture for heterogeneous multicore systems. We propose techniques to reduce the network interference effectively through MCs partition, specially designed routing algorithm, and bypass scheme for interference mitigation in heterogeneous systems.

By evaluating different CPU and GPU applications, we found that our proposed NoC architecture is able to improve the overall network performance as well as the overall system performance. Our evaluation results show that the average performance of CPU can be improved as much as 30% and GPU average performance can be increased as much as 9%, compared to the baseline 6x6 mesh network. With a specially tailored routing algorithm, our scheme can reduce over 40% links and router buffers for cost reduction. Moreover, we employ optical links in our NoC architecture to enable further energy savings. Our results show that the proposed scheme can reduce over 17% of energy consumption on the average compared with baseline heterogeneous multi-core architecture.

References

[1] L. Alhubail and N. Bagherzadeh, "Power and performance optimal noc design for cpu-gpu architecture using formal models," in *DATE*, 2019.

[2] C. R. D. Asit K. Mishra, Onur Mutlu, "A heterogeneous multiple network-on-chip design: An application-aware approach," in *DAC*, 2013.

[3] A. Bakhoda, G. L. Yuan, W. W. Fung, H. Wong, and T. M. Aamodt, "Analyzing cuda workloads using a detailed gpu simulator," in *ISPASS*, 2009.

[4] I. Bratt, "The arm mali-t880 mobile gpu," in *2015 IEEE Hot Chips 27 Symposium (HCS)*, Aug 2015, pp. 1–27.

[5] X. Bu, J. Rao, and C.-z. Xu, "Interference and locality-aware task scheduling for mapreduce applications in virtual clusters," in *HPDC*, 2013.

[6] X. Cai, J. Yin, and P. Zhou, "An orchestrated noc prioritization mechanism for heterogeneous cpu-gpu systems," vol. 65. Elsevier, 2019, pp. 344–350.

[7] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing," in *IISWC*, 2012.

[8] D. DiTomaso, A. Kodi, and A. Louri, "Qore: A fault tolerant network-on-chip architecture with power-efficient quad-function channel (qfc) buffers," in *HPCA*. IEEE, 2014, pp. 320–331.

[9] L. Gwennap, "Sandy bridge spans generations," in *Microprocessor Report*, vol. 9, no. 27, 2010, pp. 10–01.

[10] B. He, W. Fang, Q. Luo, N. K. Govindaraju, and T. Wang, "Mars: a mapreduce framework on graphics processors," in *PACT*. IEEE, 2008, pp. 260–269.

[11] H. Jang, J. Kim, P. Gratz, K. H. Yum, and E. J. Kim, "Bandwidth-efficient on-chip interconnect designs for gpgpus," in *DAC*, 2015.

[12] O. Kayiran, N. C. Nachiappan, A. Jog, R. Ausavarungnirun, M. T. Kandemir, G. H. Loh, O. Mutlu, and C. R. Das, "Managing gpu concurrency in heterogeneous architectures," in *MICRO*, 2014.

[13] K. Lee, H. Lin, and W.-c. Feng, "Performance characterization of data-intensive kernels on amd fusion architectures," in *Computer Science-Research and Development*, vol. 28, no. 2. Springer, 2013, pp. 175–184.

[14] R. Phull, C.-H. Li, K. Rao, H. Cadambi, and S. Chakradhar, "Interference-driven resource management for gpu-based heterogeneous clusters," in *HPDC*, 2012.

[15] S. S. R. G. D. R. Das, S. Narayanasamy, "Catnap: energy proportional multiple network-on-chip," in *ISCA*, 2013.

[16] J. A. Stratton, C. Rodrigues, I.-J. Sung, N. Obeid, L.-W. Chang, N. Anssari, G. D. Liu, and W.-m. W. Hwu, "Parboil: A revised benchmark suite for scientific and commercial throughput computing," *Center for Reliable and High-Performance Computing*, vol. 127, 2012.

[17] O. M. Thomas Moscibroda, "A case for bufferless routing in on-chip networks," in *ISCA*, 2009.

[18] B. Towles and W. J. Dally, "Booksim 1.0."

[19] K. Wang, A. Louri, A. Karanth, and R. Bunescu, "Intellinoc: a holistic design framework for energy-efficient and reliable on-chip communication for manycores," in *Proceedings of the 46th International Symposium on Computer Architecture*. ACM, 2019, pp. 589–600.

[20] J. Yin, P. Zhou, S. S. Sapatnekar, and A. Zhai, "Energy-efficient time-division multiplexed hybrid-switched noc for heterogeneous multicore systems," in *2014 IEEE 28th International Parallel and Distributed Processing Symposium*. IEEE, 2014, pp. 293–303.

[21] J. Zhan, O. Kayıran, G. H. Loh, C. R. Das, and Y. Xie, "Oscar: Orchestrating stt-ram cache traffic for heterogeneous cpu-gpu architectures," in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2016, pp. 1–13.