Heterogeneous Architecture Design for Mesh

Wan-Chen Yeh
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332–0250
Email: wyeh7@gatech.edu

***Abstract*—Network-on-chip (NoC) has become a popular topic in the field of Chip Multiprocessor (CMP). The core number within the network is continuously increased with the user demands for data processing. The optimization in terms of energy and throughput for a large-scale system for multiple applications executing on a single system is not a straightforward task, especially when the architecture is limited to be homogeneous. In contrast to homogeneous design, the heterogenous NoC architecture creates more possibility to improve various scenarios. In this project, several types of heterogeneous approaches are proposed. For evaluation, the average buffer utilization for each router is implemented to help the designers figure out more details on the routers. For traffic generation, the hot spot traffic pattern which is more likely happened in reality is performed as well. In addition to above functionalities, two heterogeneous designs will be demonstrated in this project. One is the multiple injection rates for a mesh topology. The nodes within a network are assigned different injection rates, based on the location of the nodes. Another one is link redistribution among diagonal nodes in mesh. According to the approaches in [1], the diagonal nodes have some characteristics so that the relevant design can improve the performance.**

1. Introduction

Multi-core architectures have been widely discussed in academia and industry due to increasing demands of data processing. Since the number of processors is continuously increasing to accommodate the emerging technologies, there have been many mechanisms to make the architectures more scalable. Most of the preceding architectures focus on *homogeneous* design. However, a great diversity of applications running on a single computation node makes the optimization more challenging because it is unlikely to have only one hardware design that is suitable for all the application sets for it. Therefore, the *heterogeneous* design starts to be attractive to people. On the other hand, the network-on-chip (NoC) plays an important role in multi-core architecture because the communication between the network nodes strongly affects the performance such as the packet latency and power consumption. Consequently, the heterogeneous NoC design becomes one of the popular topics in computer architecture.

Many innovative ideas of multi-core systems emerge from being prototyped and verified on the software simulators. In other words, the development on simulation platforms is also a critical factor. For evaluation, most of the statistics from the simulator are the overall or average values, which may not be sufficient to design the heterogeneous architectures. On the other hand, to simulate the real-world data flow, the simulators provide traffic models for designers to make their approaches more durable. One of the traffic patterns for modeling the real cases is the *hot spot* pattern that some processors will experience higher possibilities of message requests. Since heterogeneous NoC design is a relatively advanced field, the software simulators may not have relevant built-in features. Fortunately, we can still implement the novel ideas on the simulators by exploiting their flexibility and modularity.

In this project, there are several stages for implementing the heterogeneous designs. The first stage is trying to establish a set of statistical measurements for the routers instead of only reporting the global statistics. The average buffer utilization for each router will be recorded, which is very useful for discovering the distinct behaviors among the routers in the network. The second stage is to implement a traffic pattern - hot spot - in the simulator. Hot spot pattern is more close to the reality in some scenarios, which benefits the validation of novel designs. The final stage is to perform the two heterogenous scenarios. The first one is to inject the packets to the router in different rates, and compare its performance trend and buffer utilization with the baseline. The second one is to distribute different link bandwidth to the diagonal nodes in the network [1].

The remainder of this report is organized as follows: In Section 2, the fundamental concepts and related work in heterogenous NoC will be described. The proposed structure details and the implementations will be explained in Sections 3 and 4, respectively. In Section 5, the experiment methodologies and results will be demonstrated. Some details regarding to the implementation, peer reviews, and future work will be discussed in Section 6, followed by the conclusions in Section 7.

2. Background and Related Work

In computer architecture research community, the design metrics are firstly concerned about the computation. Therefore, the simulation for developing those architectures are focusing on the processors. However, as the technology of CMOS scaling advances, multi-core architectures have become a popular field because more complicated features can be realized through those architectures. As the roadmap provided by ITRS, the performance demand will grow to 300x by 2022, in other words, the number of cores within a chip will be 100x as the current design. However, the technology advance usually accompanies challenges. Unlike the single-core architectures, the evaluation for Chip Multiprocessors (CMPs) is more sophisticated because in addition to the processor cores and memory manegement, the on-chip communication (i.e. on-chip interconnect or Network-on-Chip(NoC)) is a critical factor that needs to be analyzed delicately, especially in energy consumption and latency.

In the field of NoC, the most popular topology is an $N \times N$ mesh interconnect, because it is more scalable and easier to implement on silicon. The fundamental design for NoCs uses homogeneous design, which means the routers and links in the interconnection network are all identical. Intuitively, such architectures can simplify the design progress for the network designers. There have been many approaches are using homogeneous NoC. However, one of the cons of mesh is the absence of edge-symmetry, which leads to higher traffic going through the center routers than the side routers. Moreover, the traffic patterns might vary from one workload to another, thus the router/link utilization scenarios are also divergent from one another. Therefore, the homogeneous design is not sufficient to deal with such scenarios. In contrast to the homogeneous NoC, the heterogeneous NoC has been widely explored in many different aspects such as reallocating the resources (links and buffers) for each router [1], and energy analysis on multiprocessor system-on-chip (MPSoC) for a video application is proposed in [2]. In fact, in MPSoC field, most of the SoCs are heterogeneous because the functionalities of the processing elements (PEs) are very different, and using heterogeneous NoC will benefit the complicated communication between the PEs.

To validate a novel idea of computer architecture, the features and performance of a simulation platform are very essential. One of the popular toolsets to simulate the computer architecture ideas is Multifacet General Execution-driven Multiprocessor Simulator (GEMS) [6], which provides memory system features, such as multiple cache coherence controllers. Another simulator that was developed concurrently is the M5 simulator [8], which provides comprehensive I/O models and multiple CPU models that can evaluate the novel ideas more effectively. In this project, the simulation infrastructure is gem5 [7], which is a combination of the advantages of above simulators. It provides modularity so that varieties of ISAs (Alpha, ARM, MIPS, Power, SPARC, and x86) can be applied on the CPUs and memories without additional configurations. There are two builtin interconnection network models in the Ruby memory model in gem5. The first network model is *Simple* network, which models the link bandwidth and router latency. The other one is the *Garnet* network [9], which focuses on the modeling of the router micro-architecture, including the resource utilization and flow control. In this project, the implementations and experiments are executed on Garnet 2.0 as the network model.

The traffic patterns are also an important factor for designing the NoC. The network designers usually rely on simple synthetic traffic pattern such as uniform random, bit-complement, hot spot, and tornado traffic patterns to acquire their insights into the design and stress the test. On gem5, there have been verities of traffic patterns for the users. However, the hotspot traffic has not been provided yet. Therefore, one of this project goals is to implement the hotspot traffic on gem5.

As mentioned in previous paragraphs, the heterogeneous architecture is to optimize the performance for different kinds of workloads. The first step of designing heterogeneous architecture is to figure out the resource utilization for different traffic patterns. On gem5, the statistical data will be stored in text file. The general statistics such as the average latency/hop count, the numbers of total injected/received packet, and the average virtual channel (VC) load are provided. For each memory controller of the core, the latency, bandwidth, and energy statistics are recorded. For each router on the node, the micro-architecture activity and buffer read/write status are filed as well. Although the information of the number of buffer access time is available in the statistics, the information is insufficient to help us understand the buffer utilization. In other words, a router with a higher buffer access does not necessarily mean it has a higher buffer utilization. Therefore, in this project, a newly statistical data of average buffer utilization for each router will be added. On the other hand, so far on gem5, the configurations for on-chip network are homogeneous. For example, the injection rate is the same for the whole nodes in the topology. Besides, the link bandwidth is also the same in the topology. Thanks to the flexibility and ease of implementation on gem5, we are able to make some heterogeneous extension regarding the injection rates and the link bandwidth. Moreover, we can also simulate the heterogeneous architectures along with several different configurations, and get some insight from the approaches.

3. Proposed Works

In this project, new features added on gem5 will be introduced in this section. The proposed works can be classified according to the functionality in gem5. For evaluation, the average buffer utilization for each router is implemented. For the traffic pattern, the hot spot traffic which has not been implemented is added into the ruby network. For heterogeneous configuration, multiple injection rates within mesh topology and higher link bandwidth for diagonal nodes are going to be introduced in this section.

3.1. Buffer Utilization

In gem5, the total number of buffers of each router N_{buf} can be obtained as follow:

$$N_{buf} = N_{port\_per\_router} \cdot N_{vc\_per\_vnet} \cdot N_{vnet}, \quad (1)$$

where $N_{port\_per\_router}$ is the number of ports per router, $N_{vc\_per\_vnet}$ is the number of VCs within a vnet, and N_{vnet} is the number of virtual network. The above parameters can be configured in gem5. In every execution cycle, the packets are going through the routers. By accumulating the number of occupied buffer during each cycle time, we can average the utilization by dividing the number of running cycles.

3.2. Hotspot Traffic Pattern

The traffic pattern consists of the source and destination nodes. The characteristics of a traffic pattern are affected by the message properties such as size, arrival time, and destination distribution. The destination distribution strongly depends on network topology and the task mapping onto the cores for different applications. In reality, the traffic pattern is not possible to be evenly distributed; instead, some extreme cases may result in hot spot traffic which has some nodes with more message requests then the rest. The corresponding links and buffers suffer from a large amount of data streams. Eventually, the network may become congested. Note that even a tiny imbalance happens to the traffic, the negative effects are cumulative and thus leads to congestion as well. Therefore, in this project, the hot spot traffic pattern is implemented on gem5. The model is based on the work proposed by Pfister and Norton [3], which is mentioned in [4][5]. Besides, in this project, the hot spots are located on the corner nodes. The source node N_s will generate a random number R_x, if R_x is greater than a predefined threshold R_{th}, then the destination node will be chosen from one of the hot spot nodes randomly; otherwise the destination node will be selected from the rest nodes of whole topology, according to the uniform random traffic method. The formula below shows the rule of determining the destination node:

$$N_d = \begin{cases} N_{hotspot}, \text{if } R_x > R_{th}, \\ N_{non\_hotspot}, \text{otherwise.} \end{cases} \quad (2)$$

3.3. Multiple Injection Rates

The injection rate is the probability of generating a packet for a node per execution cycle (packet/node/cycle). In other words, the behavior of inserting a packet on a node is determined by generating a random number, with the predefined precision. In gem5, the injection rate is the same for all the nodes in the mesh. However, in reality, the injection rate might be different for the nodes. In this project, the mesh will be partitioned in four quadrants, which is shown in Figure 1. The first quadrant uses a injection rate x that configured by the user. The second, third, and fourth quadrants will use $2x$, $3x$, and $4x$ of the injection rate, respectively.

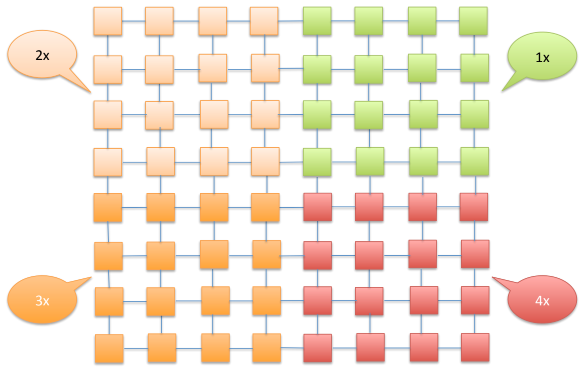


Figure 1. Multiple injection rates for a 8x8 mesh

3.4. Diagonals With Higher Bandwidth

The heterogeneous design to redistribute resources such as link and buffer are proposed in [1]. The approach proposed two kinds of router: big and small router. The big routers with more VCs and higher-bandwidth links than the small routers are placed in the diagonal nodes in mesh. This placement can make the center nodes are more capable of dealing with more traffic. Moreover, the big routers are deployed as far as possible to make each column and row has some big routers, which helps more packets going through the big routers and thus improve the performance. Also, the corner nodes will have less contention due to higher bandwidth. Therefore, in this project, a similar design is implemented: the links of the diagonal nodes will be assigned more bandwidth than the rest of the links. Instead of implement two approaches (link and buffer redistribution) at the same time, we would like to first implement only the link redistribution and verify the correctness of this approach, which is shown in Figure 2. The red links are with 2x bandwidth, while the rest links remain the same bandwidth. The experiment results will be shown in Sec. 5.2.

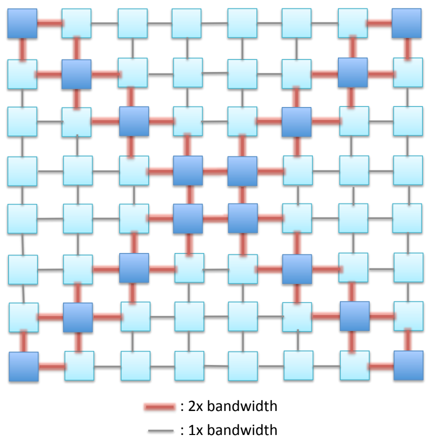


Figure 2. 2x bandwidth on diagonal nodes for a 8x8 mesh

4. Implementation

All the approaches in this project are implemented on an 8×8 mesh. The routing algorithm is XY-routing, and each router has 8 VCs. Other configurations are either using the default value provided by gem5, or changed based on the proposed architecture. The following subsections will have more details about each implementations.

4.1. Buffer Utilization

The buffers in the router can be thought as the virtual channels. Therefore, the status of buffer occupancy can be obtained by getting the VC status in the output port. In the default setting, there are five ports for each router: east, west, south, north, and local ports. For every router, there is a variable `m_buffer_util` created for recording the counts of buffer occupancy. For every cycle, the routers which have packets in the VCs will be waken up. During the wake up procedure for one router, all of the VCs in the output ports will be checked for the status which is either `ACTIVE_` or `IDLE_`. The variable `m_buffer_util` will be increased by 1 for each VC that is in `ACTIVE_` state. Note that in order to have a more precise result, we use function `is_vc_idle` provided by output unit instead of using `get_credit_count` which gets the credit from the next router. After the simulation is done, the average buffer utilization is calculated from (`m_buffer_util/execution cycle`). In *stat.txt* file, the information has been added like the following format:

```
system.ruby.network.routersxx.buffer_util
```

where `xx` is the router ID.

4.2. Hotspot Traffic Pattern

The hotspot traffic generation is a modified version of uniform random traffic. Basically one additional random number is needed for determining the destination. In this project, we just use one random number to determine whether the packet goes to the hotspots and which hotspot is the destination to save the execution time. The R_{th} is set to 52, and the random number is ranged from $[0, 100)$, which means it is a 52%-hotspot. As we mentioned in Eq.(2), the destination will be the hot spot if $R_x > R_{th}$, and the hot spot is determined by obtaining the remainder of R_x divided by 4. That is also the reason why R_{th} is selected to be 52 which can be divided by 4, making the hot spot selection with equal probabilities. The hotspot location can be selected according to the simulation requirements. In this project, since it is known that for the mesh topology, the center routers have more traffic, we select the hotspots are distributed over the four corners of the mesh to make the approach more differentiable.

4.3. Multiple Injection Rates

In every network tick, the generate packet process will be called depending on the result of the randomly generated number, as mentioned in Sec. 3.3. The network tick function is called by every router, so the injection rate can be configured in the tick function based on the router ID. Since the router IDs are in order, we can get the x and y coordinates based on the radix of the mesh. Once the coordinates is obtained, the quadrant is determined, and the injection rate can be updated.

4.4. Diagonals With Higher Bandwidth

The `bandwidth_factor` in gem5 is for setting the link bandwidth. When the link between the nodes is created, the bandwidth is then decided. Since for each `BasicIntLink` class, the router IDs for the two ends are included, we can assign different bandwidth to the links. Similar to the Sec. 4.3, the coordinates can be obtained from the router ID and radix of the mesh. We can know if the nodes are diagonals or anti-diagonals by checking the coordinates: if the x and y coordinates are the same, then it is the diagonal node; if the sum of the x and y coordinates is equal to the radix, then it is anti-diagonal. During the implementation, we found that the simulation result does not change when `bandwidth_factor` is changed due to some reasons. An alternative method to simulate this case is to change the latency of the links. This method is mentioned in [9] that links with lower bandwidth can be modeled by specifying a longer latency. Since the default latency for each link is 1 cycle, for the baseline, we assign 2 cycles of latency for all the links. In the proposed work, we assign only 1 cycle to the diagonal links and 2 cycles to the rest links.

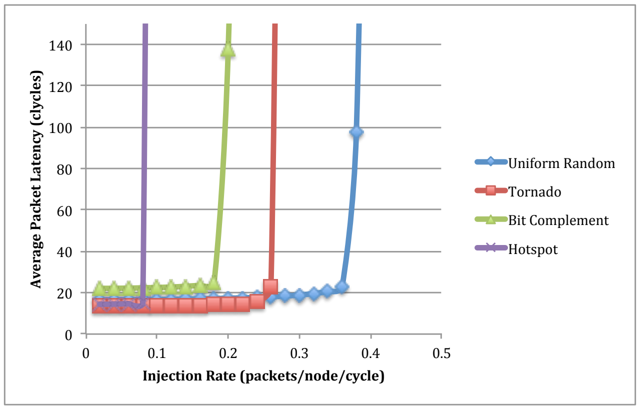
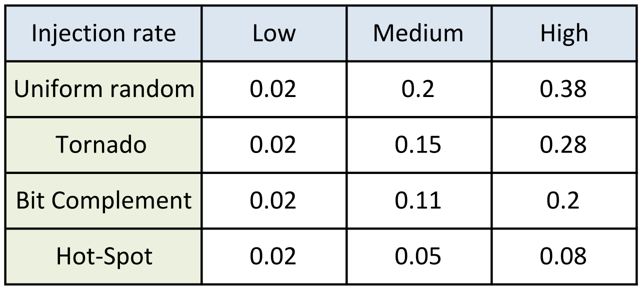


Figure 3. The injection rate v.s. latency

5. Evaluations

5.1. Simulation Methodology

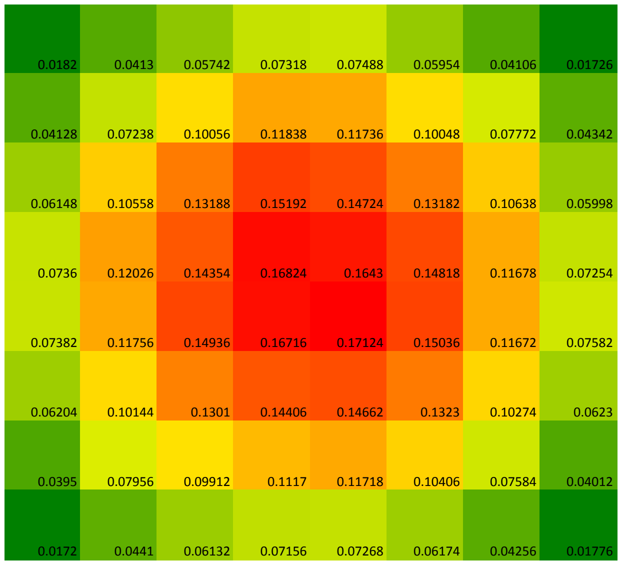
5.1.1. Baseline. The evaluations in this project will be mainly on the average buffer utilization. All of the experiments in this project have $50,000$ simulation cycles. Since in this course, we have used the *uniform random*,



Injection rate	Low	Medium	High
Uniform random	0.02	0.2	0.38
Tornado	0.02	0.15	0.28
Bit Complement	0.02	0.11	0.2
Hot-Spot	0.02	0.05	0.08

Figure 4. The level of injection rates

tornado, and *bit complement* as the traffic patterns, in this project, these patterns are the baseline traffics. For each traffic, the experiment will be run on *low*, *medium*, and *high* injection rates. The level of injection rate is depending on the result of injection rate v.s. latency relationship. The high injection rate is happened at the saturation point, which a has relatively high latency. Figure 3 illustrates the relationship between injection rate and average packet latency, and we select several levels of injection rate as shown in Figure 4.



0.0182	0.0413	0.05742	0.07318	0.07488	0.05954	0.04106	0.01726
0.04128	0.07238	0.10056	0.11838	0.11736	0.10048	0.07772	0.04342
0.06148	0.10558	0.13188	0.15192	0.14724	0.13182	0.10638	0.05998
0.0736	0.12026	0.14354	0.16824	0.1643	0.14818	0.11678	0.07254
0.07382	0.11756	0.14936	0.16716	0.17124	0.15036	0.11672	0.07582
0.06204	0.10144	0.1301	0.14406	0.14662	0.1323	0.10274	0.0623
0.0395	0.07956	0.09912	0.1117	0.11718	0.10406	0.07584	0.04012
0.0172	0.0441	0.06132	0.07156	0.07268	0.06174	0.04256	0.01776

Figure 5. Heat map for uniform random traffic (low injection rate)

5.1.2. Average Buffer Utilization. For the evaluation, the average buffer utilization for each router is displayed in a heat map as shown in Figure 5. The value in each rectangle represents the average buffer utilization per cycle. We use graded color scale to show the relative buffer occupancy: the red means higher usage in buffer, the yellow means medium usage, and the green means lower buffer usage. The router order is based on that defined in gem5: router 0 is at the bottom left corner, and router 63 is at the upper right corner.

5.1.3. Area and Energy Evaluation. In the Section 5.2.4, we will evaluate the performance by using DSENT [10] to obtain the area and read/write energy of the buffer. First, according to our approach mentioned in Section 3.1, the average buffer utilization per cycle time will be obtained for each router. Secondly, we select one of the center routers for convenience from the report given by executing DSENT, and obtain its information of buffer area and the read/write buffer energy. Finally, the sum of average buffer utilization of every router multiplied by the area per buffer or energy per buffer. Note that the area and energy from DSENT is for one router, so the values need to be divided by the total number of buffers for a router.

5.2. Results

5.2.1. Baseline. According to the injection rate levels as shown in Figure 4, we simulate the average buffer utilization for the baseline traffics as shown in Figures 6, 7, and 8. Each figure has three sub figures which represent the low, medium, and high injection rates, from left to the right.

In uniform random traffic, we can observe that the relative occupancy for low and medium injection rates are similar to each other. However the average utilizations are 0.17 and 5.43 for low and medium injection rates, respectively. In other words, the buffer usage in the center routers for medium injection rates are approximately 32x than the low injection rate. At high injection rate which the network starts to have more congestion, the average buffer utilization along x-direction is more noticeable because we use XY-routing algorithm for the experiments.

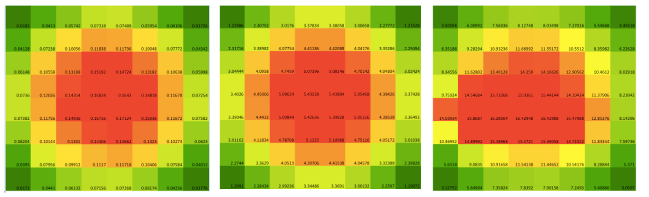


Figure 6. Heat maps for uniform random traffic

The result of tornado traffic is shown in Figure 7. The tornado traffic implemented in gem5 focuses on the x-direction, and the y-direction remains the same as the source router. Therefore we can see that the relative buffer utilization along y-direction is inconspicuous, while the relative buffer occupancy for x-direction is more notable. Conceptually the buffer utilization should be the same on the x-direction according to the rule of tornado traffic generation that the displacement from the source to the destination is the same. The uneven distribution is due to the topology limitation from mesh as we have mentioned in Section 2. Similar trend as the uniform random traffic can be found in bit complement traffic, which is illustrated in Figure 8. At high injection rate, the utilization tends to distributed along the x-direction because of XY-routing and the traffic generation rule.

5.2.2. Hot Spot Traffic. The simulation results are shown in Figure 9. The buffer occupancy is higher for the routers in the same x-axis as the hot spots. This is due to the routing

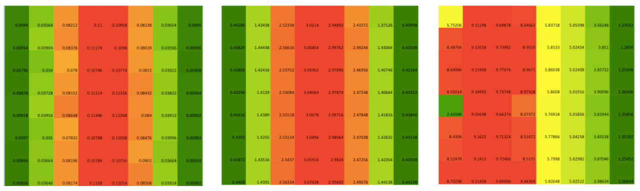


Figure 7. Heat maps for Tornado traffic

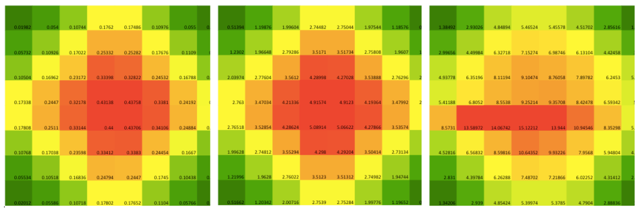


Figure 8. Heat maps for Bit Complement traffic

algorithm that all the packets whose destinations are hot spots tend to move to the left or right edge routers at first. To mitigate this issue, a optimization for the routing algorithm should be considered in this design.

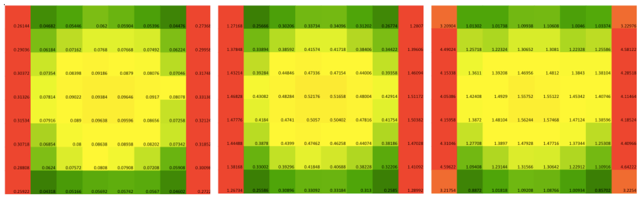


Figure 9. Heat maps for Hot Spot traffic

5.2.3. Multiple Injection Rates. The experiments of multiple injection rate is shown in Figure 10. In this experiment, we use uniform random traffic with injection rate 0.02, 0.2, and 0.38 for low, medium, and high injection, respectively. As we introduced in Section 3.3, we can observe the average buffer utilization is as expected. The occupancy on the boundaries between the quadrants might be affected by its neighbors. For example in quadrant II, the utilization is lower at the boundary to quadrants I, while more traffic passing through the boundary to quadrant III. At high injection rate, the boundaries become obscure because network congestion occurs more frequently.

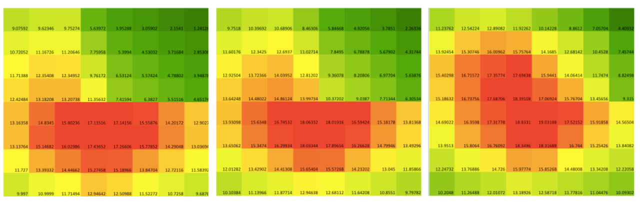


Figure 10. Heat maps for multiple injection rates

5.2.4. Diagonals with Higher Bandwidth. Similar to the methodology mentioned in Section 5.1.1, we run several simulations to obtain the saturation points for this approach. The result is shown in Figure 11. We can observe some improvements from this approach. First, the average packet latency when we apply moderate injection rates is improved for all of the three baseline traffics by using 2x bandwidth at the diagonal links. Secondly, the saturation point for uniform random traffic is increased from 0.36 to 0.38, which means the proposed work has more capability for the traffic.

In this experiment, we modify the latency of each link to simulate different link bandwidth. Therefore, we have another baseline experiment that the latency for each link is 2 cycles, as shown in Figure 12. The proposed diagonal deployment is shown in Figure 13. We select the uniform random as the traffic pattern. We can observe that for the medium injection rate in our approach, the four center routers have a relatively low buffer utilization. In order to have a more clear observation of the improvement, Figure 14 shows the improvement in terms of percentage of buffer utilization. The negative values in the routers represent the buffer occupancy is reduced. As we can see, almost all of the routers has lower buffer utilization when diagonal links have 2x bandwidth. For low and medium injection rates, the utilization among the diagonal routers is reduced by 50% in average (up to 57% at the center routers). Even when high injection rates, the average utilization can be reduced by 30% in average.

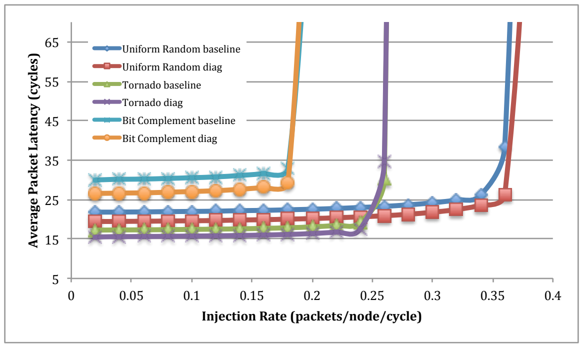


Figure 11. The injection rate v.s. latency for 2x bandwidth on diagonals

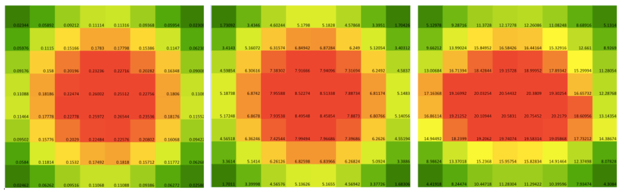


Figure 12. Heat maps for uniform random traffic, link latency = 2

Figures 15 and 16 present the evaluation of the area and energy. We can observe that with our diagonal deployment, the area and energy are greatly reduced. For the buffer area, it is reduced approximately by 17%, 16%, and 12% for low, medium, and high injection rates, respectively. On the other

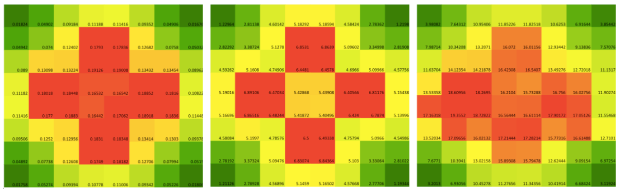


Figure 13. Heat maps for higher bandwidth links on diagonals

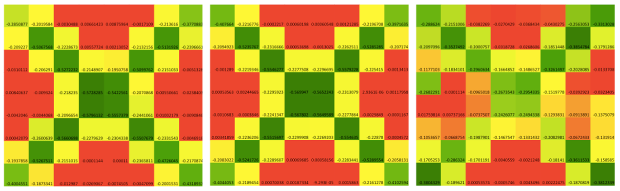


Figure 14. Heat maps for the improvement on higher bandwidth on diagonals

hand, the buffer read and write energy can be reduced by 9% and 12%, respectively, for high injection rate (up to 17% when low injection rate).

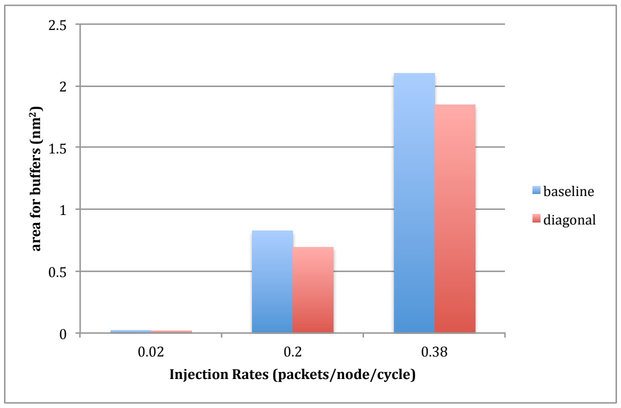


Figure 15. Buffer area evaluation

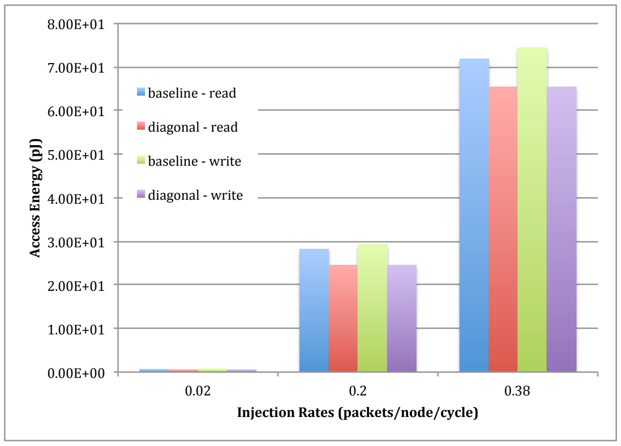


Figure 16. Buffer energy evaluation for read and write

6. Discussions and Future Work

6.1. Peer Review Feedback

I appreciate the peers' valuable suggestions for this work, and would like to summarize those feedbacks as follows:

- The simulation results are provided in the final report with the some interesting observation and discussion. For example, the proposed work of diagonals with 2x bandwidth improves the saturation point and reduces the latency, and these facts are explained with the simulation results.
- The implementation details should be more concise and precise to avoid any confusion to the readers. For instance, the approach to obtain buffer utilization is rewritten in a more clear way to make the readers understand how it is implemented.
- For the hot spot implementation, the motivation or introduction should be more organized to convince the readers that this pattern is useful in some specific scenarios. For example, I have added some description about how the hot spot is determined and why we choose the corner routers as the hot spot in Section 4.2.
- Although all the experiments are executed in an 8×8 mesh, the implementations regarding the gem5 source code such as hot spot, multiple injection rates, and diagonals with 2x bandwidth have been considered the scalability for different sizes of mesh. In other words, there is no additional modification needed for executing other size of meshes. Due to the time limitation, the results for different sizes of meshes are not able to be provided. However, those simulations are executable for sure and we still can obtain the results and have some observation of the experiments.

6.2. Future Works

In this project, we have some assumptions and predefined parameters. For average buffer utilization, the buffer depth for each VC is one-flit, which simplifies the scenario and the implementation. The implementation will be more compatible to more complicated cases if the buffer depth is considered as well. For Hotspot traffic pattern, the locations of the hot spots and R_{th} are pre-determined for convenience. Besides, in multiple injection rates, the injection rate for each part is a multiple of the one configured in the run time. The above pre-defined parameters can be modified to be user-defined in the future to make the simulations easier to execute.

7. Conclusion

In this project, the proposed works are implemented on gem5, and all of them are related to the heterogeneous design. The first step to design the architectures with heterogeneous routers is to have some observations from the simulation statistics. Therefore, the average buffer utilization is implemented to make the designers have more insights about heterogeneous design. In addition to the evaluation feature, the hot spot traffic pattern is established to make further novel architecture design more comprehensive. Moreover, multiple injection rates within a mesh is performed, and diagonal nodes with higher bandwidth is also completed. The heterogeneous design is an inevitable trend on NoC, since the optimization methods for the applications running on the same machine are adaptive and affected by many different aspects.

References

[1] A. K. Mishra, N. Vijaykrishnan and C. R. Das, ”A case for heterogeneous on-chip interconnects for CMPs,” *Computer Architecture (ISCA),* 2011 38th Annual International Symposium on, San Jose, CA, 2011, pp. 389-399.

[2] A. Jarrah and M. M. Jamali, ”Energy analysis and NoC design for heterogeneous MPSoC platform for a video application,” *2013 IEEE 56th International Midwest Symposium on Circuits and Systems (MWSCAS),* Columbus, OH, 2013, pp. 437-440.

[3] G.F. Pfister and V.A. Norton, ”Hot Spot Contention and Combining in Multistage Interconnection Networks”, *IEEE Trans. on Computers,* vol. 34, no. 10, pp. 943-948, 1985.

[4] M. M. H. Rahman, Y. Inoguchi, Y. Sato, Y. Miura and S. Horiguchi, ”On hot-spot traffic pattern of TESH network,” *Computer and Information Technology,* ICCIT 2008. 11th International Conference on, Khulna, 2008, pp. 359-364.

[5] M. M. H. Rahman, M. A. H. Akhand, Y. Miura and Y. Inoguchi, ”Hot-Spot Traffic Pattern on Hierarchical 3D Mesh Network,” *Advanced Computer Science Applications and Technologies (ACSAT),* 2014 3rd International Conference on, Amman, 2014, pp. 56-60.

[6] Milo M. K. Martin, Daniel J. Sorin, Bradford M. Beckmann, Michael R. Marty, Min Xu, Alaa R. Alameldeen, Kevin E. Moore, Mark D. Hill, and David A. Wood, ”Multifacet’s general execution-driven multiprocessor simulator (GEMS) toolset,” *SIGARCH Comput. Archit,* News 33, 4 (November 2005), 92-99.

[7] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood, ”The gem5 simulator,” *SIGARCH Comput. Archit,* News 39, 2 (August 2011), 1-7.

[8] N. L. Binkert, R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi and S. K. Reinhardt, ”The M5 Simulator: Modeling Networked Systems,” in *IEEE Micro,* vol. 26, no. 4, pp. 52-60, July-Aug. 2006.

[9] N. Agarwal, T. Krishna, L. S. Peh and N. K. Jha, ”GARNET: A detailed on-chip network model inside a full-system simulator,” *Performance Analysis of Systems and Software,* 2009. ISPASS 2009. IEEE International Symposium on, Boston, MA, 2009, pp. 33-42.

[10] C. Sun et al., ”DSENT - A Tool Connecting Emerging Photonics with Electronics for Opto-Electronic Networks-on-Chip Modeling,” *Networks on Chip (NoCS),* 2012 Sixth IEEE/ACM International Symposium on, Copenhagen, 2012, pp. 201-210.