Intelligent Spatial-based Resource Allocation Algorithms in NoC

Jingling Yuan
Computer Science and Technology School, Wuhan University of Technology, Wuhan, China
Email: yjl@whut.edu.cn

Xin Fu, Tao Li and Minlong Yang
Dept. of Electrical and Computer Engineering, University of Kansas, Kansas,USA
Dept. of Electrical Engineering and Computer Science, University of Florida, Florida, USA
Computer Science and Technology School, Wuhan University of Technology, Wuhan, China
Email: xinfu@ittc.ku.edu, taoli@ece.ufl.edu, yamilo123@126.com

***Abstract*—The Network-on-Chip (NoC) approach is a promising solution to the increasing complexity of on-chip communication problems due to its high scalability. A NoC architecture design with ultra-low latency and high throughput is critical in order to support a wide range of applications. In this paper, we propose novel spatial-based NoC resource allocation algorithms to reduce the communication congestions so that efficiently improve the overall performance. We propose both offline (e.g. probability-based searching, genetic-based searching) and online (e.g. online greedy, one-corner, four corner) algorithms for NoC resource allocation. Based on our experiment results, the proposed algorithms show strong capability of improving the NoC performance.**

***Index Terms*—Network-on-chip, multi/may-core processors, NoC resource allocation, 2D compaction algorithms, spatial-based global optimization**

I. INTRODUCTION

Multi-core processors represent a revolution in computing technology. Processors with two, four and eight cores have already entered the market. Processors with tens or possibly hundreds of cores may be a reality within the next few years (e.g. Intel Teraflop research chip has 80 cores [1]). To ensure that multi-core performance will scale with the increasing number of cores, network-on-chip (NoC) architectures [2, 3] are increasingly being deployed in hardware design. The trend toward many-core processors will result in sophisticated and large-scale NoC substrates with self-contained hardware components (e.g. network interface and router) proximate to the individual cores but globally distributed across the entire chip. As the number of cores increases, a NoC architecture providing low latency and high throughput is desirable. With the increasing number of heterogeneous applications/threads executing concurrently in future many-core processors. NoC utilization is becoming less predictable and exhibiting irregular spatial diversity. Because one application may use the spatially distributed NoC components significantly different than that of another application. When multiple applications run simultaneously in the many-core processor, traffic interference among those applications will largely degrade the NoC performance if they are not properly mapped across the chip.

Although examining the behavior of one NoC component may be sufficient for optimizing the performance and efficiency of individual cores, to achieve global and cooperative resource allocation in many-core environments, we need NoC management techniques that can adapt to the complex spatial traffic patterns among workloads as well as their interactions beyond the scope of individual cores and NoC components. In this paper, we proposed spatial-based global and cooperative NoC resource allocation schemes to achieve this goal. Since the scalability and sustainability of future many-core processors crucially depends on efficiently and intelligently managing an increasingly large and complex hardware investment, we expect that the proposed novel techniques will open the door for a new class of optimizations that cannot be effectively achieved using conventional methods.

The contributions of this paper are:

1) We proposed two spatial-based (probability-based searching and genetic-based) offline NoC resource allocation techniques that can cost-effectively reduce NoC traffic interference among concurrently running applications. Probability-based searching offline algorithm is simple, easy to implement and quite efficient to solve optimization problems. Genetic Algorithm is also an efficient way of solving problems using the same combination of selection, crossover and mutation to explore a global solution to a problem.

2) We further explored spatial-based on-line congestion avoidance techniques to achieve the optimal NoC throughput at rut-time. We used online 2D compaction algorithms: one-corner compaction, in which we compacted the workloads in 2D towards one corner of the chip; and four-corner compaction, in which workloads close to a certain chip corner are compacted towards that corner using the one-corner compaction algorithm. Corner-based compaction allows local congestion to be dispersed without interfering with the

doi:10.4304/jcp.8.3.541-549

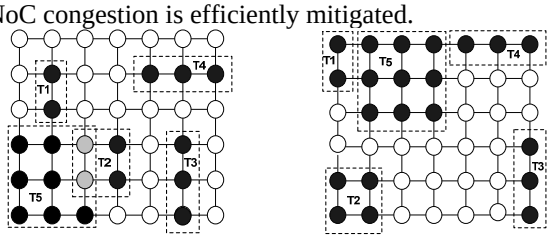
rest of the chip area [4]. Therefore, the probability that global congestion will occur is reduced. In addition, the increased contiguous space released by compaction will make it easier to accommodate new incoming workloads.

II. 2D Spatial NoC Traffic Interference

In NoC, packet flows originating from multiple cores compete for resources while traversing the output link at each intermediate router along their paths. The congestions within routers and across links spanned along the shared paths strongly affect the overall performance of the NoC using deterministic or adaptive routing algorithms [5-7]. With the increasing number of cores on a chip, flow congestion control is imperative to ensure fair and efficient NoC resource sharing among all cores. The efficiency and scalability of NoC in many-core processors will crucially depend on the congestion control capability. There has been significant work on NoC flow control [8-15]. Most of the current NoC congestion control approaches used local information, such as the usage of buffer associated with an output port in the router, as a measure of communication traffic. An efficient technique which is consisting of region selection and node allocation to allow for new applications to be added to the system with minimal interprocessor communication overhead is studied [16]. A methodology for mapping multiple use cases onto NoCs, where each use case has different communication requirements and traffic patterns is proposed [17]. Dynamic task mapping scheme in NoC-based heterogeneous MPSoCs is used for the channel load minimization for improving the performance [18]. Prior studies [5] observed that incorporating regional congestion information from neighboring routers into adaptive routing can improve NoC efficiency. For the large-scale NoC in many-core processors, the gathered local/regional congestion knowledge may not represent the global traffic status. Better utilization of NoC resources can be achieved if the spatially distributed congestion status can be exploited to implement global and cooperative congestion control strategies.

A many-core processor can execute a large number of workloads in parallel. Each workload may contain multiple concurrently running threads that request communication. They are placed on cores that are spatially close to each other in order to reduce the number of hops that flits traverse in NoC and hence, the network latency. Generally, OS will randomly select a region which is composed of a number of contiguous cores for the workload execution. Since workloads are randomly assigned to a region, several cores are shared by more than one workload. As a result, their attached routers will deal with communication requests from multiple workloads which cause high traffic congestion and greatly degrade the performance. The degree of workload overlapping at cores significantly affects the NoC network latency. Moreover, the entire NoC resources are fragmented while placing workloads into the many-core processor. High fragmentation can lead to undesirable situations where new workloads cannot be placed efficiently to avoid the workload overlap in cores. Therefore, a higher degree of workload overlapping occurs which further exacerbates the latency.

In this paper, we proposed novel techniques to achieve global and cooperative congestion control across the NoC architecture. We explored both offline and online 2D compaction mechanisms to schedule applications to different cores/locations so that the total number of congested areas can be minimized or distributed uniformly across the entire NoC. Fig. 1 compares the global NoC congestion under (a) the random resource allocation and (b) our proposed mechanisms (e.g. four-corner-based compaction) when five workloads (T1~T5) are assigned to the processor. As Fig. 1 shows, under the random resource allocation, a large number of cores are shared by the workloads showed in Fig. 1(a), while our technique efficiently avoids the overlap and increases the maximum idle region showed in Fig. 1(b), therefore, the NoC congestion is efficiently mitigated.



(a) Random (b) Four-corner-based

Figure 1. NoC congestion under (a) random and (b) spatial-aware (four-corner-based) resource allocations

III. The Proposed Spatial-Aware Congestion Avoidance Techniques

In this section, we proposed both off-line and on-line techniques that aim to minimize or completely avoid NoC contentions among concurrently running applications. They are targeting at two different scenarios: (1) when the number of workloads scheduled to the processor and each workload's resource requirements are pre-known, we can apply the off-line 2D compaction algorithms to perform the global congestion avoidance through all of the workloads and obtain the optimal solution for the resource allocation; (2) when both the quantity and resource requirements of the incoming workloads vary dynamically, and OS is blind to the their characteristic, the off-line mechanism is not feasible. We resort to the online algorithm computation when every workload is inserted. It is simple, fast and able to obtain the local optimal resource allocation to the new arriving workload that could finally lead to the near-optimal/optimal solution to all of the inserted workloads.

Generally, when scheduling multiple threads of a workload into the many/multi-core processor, OS will assign each single thread to the core with the minimum distance to cores on which other threads have heavy communications with it. By doing that, the number of hops for packets travelling among the cores could be efficiently reduced to enhance the performance. In our paper, we assumed that the core allocation to each thread

in the same workload is pre-defined based on their communication characteristics. In another word, the shape of the region that a workload occupies in the processor will keep the same no matter the random allocation or the techniques applied. Our congestion avoidance algorithms mainly focus on moving and rotating the region to achieve the optimal resource allocation. As discussed in Section 2, both the workload overlap and the maximum free region affect the NoC congestion. Our proposed congestion avoidance algorithms consider the optimization of following two metrics: 1) the degree of overlapping, which can be expressed as formula (1)

$$\sum_{i=2}^{\infty} (n * 2^{i-2}) \quad (1)$$

Where i is the number of applications and n is the number of overlapped modules and 2) the maximum free region, which is defined as a contiguous area that contains the maximal number of idle cores. Intuitively, a good congestion avoidance algorithm is capable of minimizing the degree of overlapping while maximizing contiguous white space simultaneously. Alternatively, one can assign a higher priority to reduce overlapped area.

A. Off-line Techniques

In an off-line scenario, one can afford to spend more time to explore optimal or near-optimal solutions. In this case the problem is similar to the traditional packing problems. We proposed two off-line congestion-aware algorithms based on probability-based searching algorithm and Genetic algorithm respectively.

1) Probability-based Searching Algorithm

Traditional greedy algorithm follows the problem solving with meta-heuristic which makes local optimum choice at each stage with the hope of finding the global optimum solution. However, it may not be able to obtain the global optimum resource allocation in our study because the greedy algorithm will not relocate or rotate the applications previously allocated in the processor when assigning the resource to a new application at each stage. In this paper, we proposed a dynamic probability-based searching algorithm which extends from the greedy algorithm. In this algorithm, we assume the workloads are assigned sequentially. When a new application requests for resource allocation, our probability-based algorithm will dynamically generate a percentage value for each previous assigned workloads, the value describes the probability of moving or rotating the workload. During the workloads' relocation and rotation, our algorithm will search for the optimum solution. Generally, the larger number of workloads with high probability to move/rotate, the higher chance but longer time to find the global optimum solution. In order to achieve the optimum trade-off between execution time and finding the best solution, only two workloads will be given high probability of relocation/rotation. The effectiveness of a new solution is evaluated using the above two metrics: the degree of overlapping and the maximum free region. Fig. 2 describes the details of probability-based searching off-line congestion-aware algorithm and Fig. 3 illustrates an example of applying this algorithm to optimize the global placement of 5 applications across a 6×5 NoC substrate.

Step 1. Obtaining the applications' initial application placement. We first randomly place all the applications across the NoC substrate. As shown in Fig. 3(a), there are five applications and three of them are overlapped based on the random allocation.
Step 2. Separating the area overlapping and connection based on the initial state. Fig. 3(b) shows a temporary solution when rearranging these five applications.
Step 3. Getting a new solution based on temporary solution. Randomly perform movement and rotation on some applications in certain probability while avoiding or minimizing overlap between applications' NoC traffic footprints. Fig. 3(c) shows one of the candidate solutions.
Step 4. Finding the best solution among the candidate solutions by comparing their maximum free region. If a new solution is better than the current solution, the new solution is selected. Otherwise, the new solution is given up and the algorithm returns back to step 3. Such as Fig. 3(d) illustrates the optimum solution.

Figure 2. The probability-based searching off-line congestion avoidance algorithm

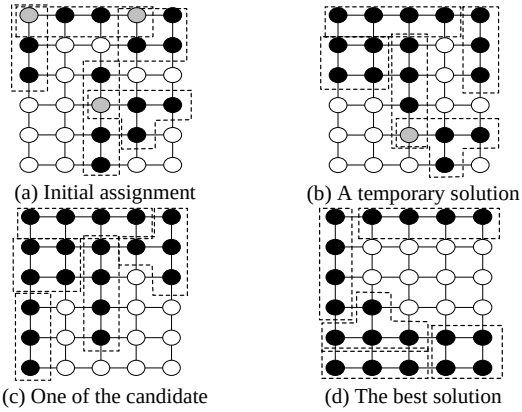


Figure 3. An example of applying probability-based searching algorithm

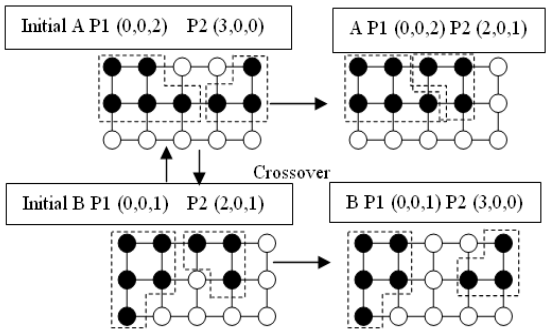
2) Genetic-based Algorithm

A genetic algorithm (GA) is a global search heuristic technique, which uses mechanisms inspired by evolutionary biology such as mutation, selection and crossover, to find optimum/near-optimum solution to the search problems. A typical genetic algorithm requires a genetic representation of the solution domain and a fitness function to evaluate the solution domain. Our proposed GA-based algorithm changes the location of basic point (We define the left-top point of an application's NoC footprint as the basic point in our paper) and applies evolutionary operations to obtain the optimal solution. Our algorithm used a set of positions and states of basic points as the genetic representation of the solution domain.

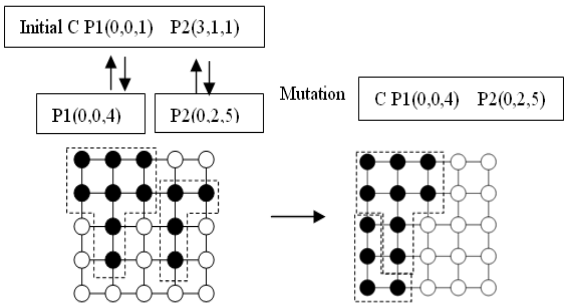
As an example, Fig. 4(a) shows a crossover process between two applications T1 and T2 using two initial states A and B. Fig. 4(b) illustrate the mutation process based on an initial state C. In Fig. 4, the application's basic point is represented as P(x, y, z), where x, y are coordinate of the point and z represents the state (0: initial state, 1: a state after 90 counterclockwise rotation,

2: a state after a 180 counter-clockwise rotation, 3: a state after a 270 counter-clockwise rotation, 4: a state after a symmetrical rotation via y=0, 5/6/7: states after 90/180/270 counter-clockwise rotation from state 4).

Note that the degree of overlapping and the maximum free region are also used as two metrics to find the best solution. The maximum free region is used as the fitness function to evaluate the solution domain when the degree of overlapping is minimum. Fig. 5 describes the details of GA-based off-line congestion-aware algorithm.



(a) Crossover Procedure (P1 and P2 are the basic points of applications T1 and the T2)



(b) Mutation Procedure (P1 and P2 are the basic points of the applications T1 and T2)

Figure 4. An illustration of crossover and mutation operations in the proposed GA based algorithm

Step 1. Building up initial population which is composed of all basic points' coordinate data P(x, y, z) of different application's NoC traffic footprint, which is the sub-space of solution space.
Step 2. Using minimal degree of overlapping D and maximum free region as constraint functions F to compute the fitting value of each basic point coordinate data in the population.
Step 3. Assuming that the objective function is global maximum free region when the degree of overlapping D is minimal and the maximal number of generation is MAX (e.g. MAX=1000), if F(P1)>F(P2), then coordinate data P1 is better than P2, else go to step 2 to select a better solution.
Step 4. For every iteration, the selected function will choose better coordinate data as the parent class from the population. A crossover operation of the two selected parents is performed based on the crossover rate. Coordinate data of new population is then mutated based on mutation probability.
Step 5. As the number of iterations increases, the excellent coordinate data will dominate the population. Finally the best solution will be produced.

Figure 5. GA-based off-line congestion-aware algorithm.

B. On-line Algorithm

Different from the off-line algorithm, online placement algorithms can arrange applications at run-time. In order to perform the online placement, the algorithm is required to quickly find an appropriate empty area for the new arriving application and reduce the overlapping without knowing the characteristics of the following incoming applications. In other words, our goal is to dynamically place the application based on its NoC traffic characteristics and meanwhile achieve the optimum/near-optimum resource allocation to all of the inserted applications. The methods should be fast, pack the modules tightly on the layout and efficiently avoid or minimize congestion. The three online congestion-aware algorithms are proposed as follows.

1) Greedy-based Algorithm

The greedy-based online algorithm can minimize the internal communication cost of running applications to reduce performance overhead without knowing the next application in advance. The maximum free region and degree of overlapping metrics are identical to the off-line Probability-based searching algorithm proposed in Section III. A. 1), and the basic flows of the two algorithms are similar as well, but the probability-based location selection policy applied in the online algorithm is greatly different. First, since our online algorithm has no clue about the quantity and the characteristics of all the inserted applications, there is no "initial application placement" which is the starting point of the off-line algorithm process. Moreover, the online algorithm focuses on the speed of finding an appropriate empty area for the arriving application, and every step solution is treated as the current best solution. In one sentence, the offline greedy-based algorithm takes the global optimal solution as the most important factor and performs global and recursive placement, its execution time is significantly longer than that of the online greedy-based algorithm.

2) Corner-based Algorithms

Corner-based online algorithm is a two-dimensional compaction algorithm: we compact application's NoC traffic footprint in two dimensions towards corners of the chip to release space for new incoming applications. There are two major compact methods. The basic one is called one-corner, it compacts application's NoC footprints towards one corner of the chip. The extended one is named as four-corner, it compacts application's NoC footprints towards four different corners instead of only one corner in the one-corner-based algorithm. The maximum free region is also used as constraint to select the best solution. These algorithms effectively reduce area fragmentation and therefore, improve the overall performance.

- One-corner Algorithm

In the one-corner algorithm, every corner can be selected as the placement corner. The applications are moved in both vertical and horizontal directions to one of the corners (e.g., southwest, southeast, northeast, or

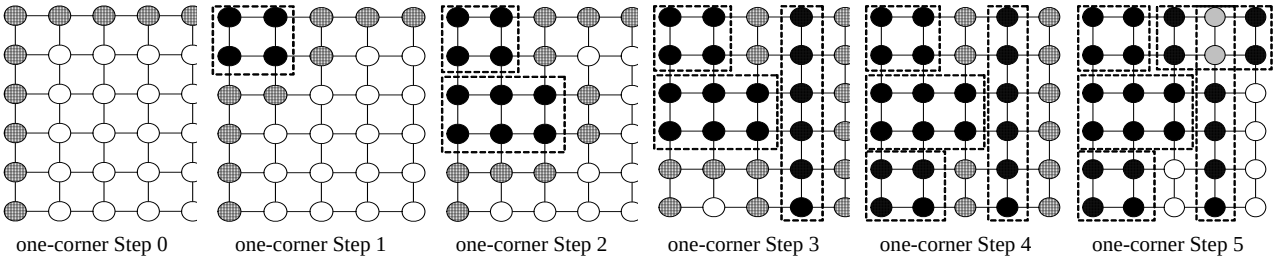


Figure 6. Example of congestion-aware algorithm based on one-corner algorithm

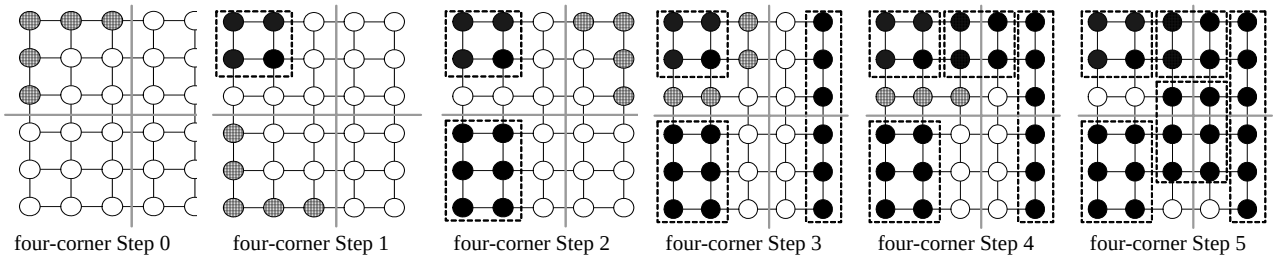


Figure 7. Example of congestion-aware algorithm based on four-corner algorithm

northwest) of NoC substrate. In this paper, we used the northwest corner so all the incoming applications will be placed to the same corner one by one based on the placement strategy. The goal of our algorithm is to find appropriate location to place the incoming applications. In our paper, the cores at the boundary of previously inserted applications are defined as the location edge points. They will be selected as the basic point of the next application to be placed.

Fig. 6 shows an example of the one-corner algorithm (assuming the north-west corner as the placement corner). All applications will be placed into the same corner.

- Four-corner Algorithm

The four-corner algorithm is derived from the one-corner algorithm. The difference is that the layout is divided into four regions which are the northeast, the northwest, the southeast, and the southwest. Placement depends on these four different areas and every area has its own maximal regional free space. Edge points are also defined as the next corner edge and the placed applications' boundary in this placement corner, which can be selected as the position of the following application' basic point. In order to make the full utilization of the four corners and keep the maximum free region , the corner with the maximal regional free space has higher priority to be selected as the placement corner. Therefore, the first step in our four-corner algorithm is to select the appropriate placement corner according to the maximal regional free space. When the four-corner show the same free space, applications can be assigned to the corner by the placement order that is northwest corner, northeast corner, southeast corner and southwest corner. Fig. 8 illustrates the algorithm in details.

Fig. 7 shows an example of the four-corner online algorithm. Grid units are the edge points which are selected as the basic point of the next application to place.

Corner based algorithm is not only simple but also allows local congestion to be dispersed without interfering with the rest of the chip area. Therefore, the global congestion will be reduced. In general, four-corner algorithm will get the better placement because it improved one corner and distributed contention by area partitioning.

Step 1 Dividing the layout into four corner area.
Step 2 Determining which corner area the application's NoC footprint should be placed according to the maximal regional free space and the placement order (northwest, northeast, southeast, southwest). Selecting appropriate edge points as the basic point of the new arriving application's NoC footprint
Step 3 Filling this application's NoC footprint when the position of its basic point is confirmed. Repeat step 2 and step 3 until all application's NoC footprints are filled.
Step 4 Optimize all application's NoC footprints using maximum free region as constraint function.

Figure 8. Four-corner based algorithm

C. Analysis on Online and Offline Algorithms

The major difference between offline and online algorithms is: the offline algorithm aims to achieve the global optimum resource allocation; while the online algorithm focuses on the execution time, it achieves the local optimum solution when an application is assigned to the processor. In general, offline solutions will be the optimal standard for online algorithms.

We compare the online with offline algorithms by assigning 16 applications to the 30-node mesh network. Detailed experimental setup is presented in Section 4. The Fig. 9 compares the effectiveness of both online and offline algorithms in achieving the optimum resource allocation. X-axis represents the layout testing cases (the X-axis represents the number of applications, the number i layout comes from adding an additional application to the number i-1 layout). Y-axis represents the effectiveness of the algorithms. The effect of mapping layout is divided into two aspects: the maximum free

region and the degree of overlapping. When y is larger than zero, it represents the maximum free region with no overlapping; to the contrary, when y is smaller than zero, it describes that the region contains overlap, and the larger absolute value of the negative data means a higher degree of the workload overlapping. Interestingly, as Fig. 9 shows, offline and online algorithms show similar capability in effective resource allocation. It is because the benefit of offline algorithm is significantly affected by the scale of the searching area (i.e. the number of cores in the processor). In the processor with few cores, the local optimum solutions obtained by the online algorithm is close to the global optimum solution generated by the offline algorithm since there are limited number of choices for offline algorithm to select the best resource allocation .

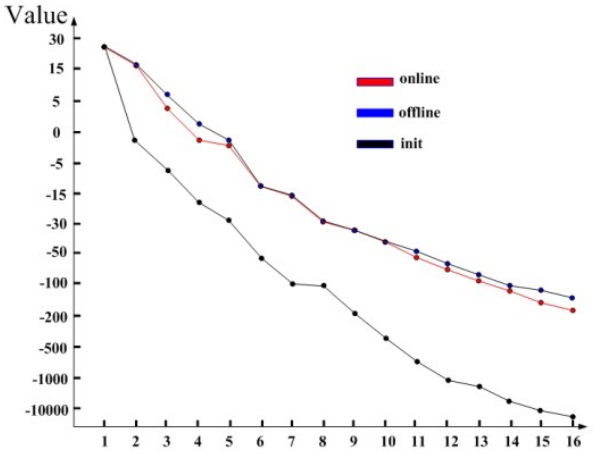


Figure 9. Effectiveness comparison between online, offline algorithms and init (with no optimization)

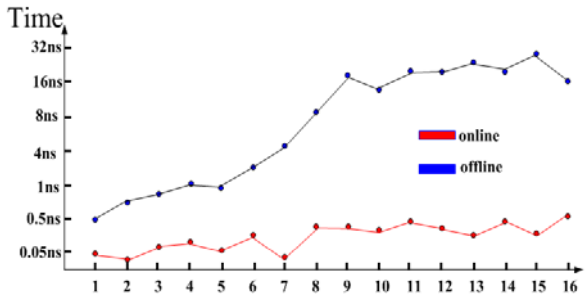


Figure 10. Execution Time comparison between online and offline

Fig. 10 presents the execution time of offline and online, as it shows, as the number of applications increases, the offline algorithm requires longer time to obtain the global optimum solution, while execution time of the online algorithm maintains low since it only consider the current inserting application, and achieve the local optimum solution.

IV. Experimental Methodology

In our study, we used Garnet [19], which is a detailed cycle-accurate NoC simulator, and extend it to support both dimension-order routing (DOR) and the regional congestion aware adaptive routing [5]. All simulations are performed for a 30-node (6×5) mesh network. We restricted our evaluation to 2D mesh NoC, but the general principles presented here could be applied to other NoC topologies as well. Each VC group has 4 virtual channels. Each VC holds four 128-bit flits. One packet consists of either one or five flits. Since workloads such as SPLASH-2 do not actively exercise the large scale NoC architecture, we opted to evaluate our techniques using synthetic workloads. We performed experiments at 200 chips.

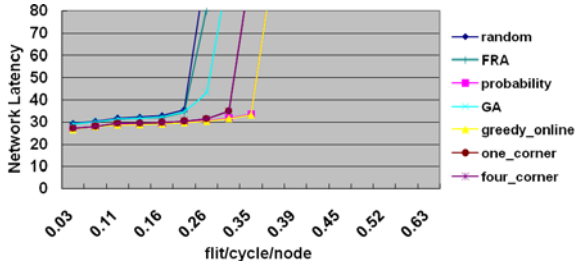
At each chip, we first generated a random variable to represent the number of workloads that will be assigned to it. We set the maximum number as 16 in our simulated 30-node network since allocating an extremely large number of workloads could saturate the NoC without any proceeding. On the other hand, the minimum number is set as 2 because very few workloads hardly cause the overlap and performance degradation in NoC. We further generated a variable for each workload to describe the number of cores required by the workload. The workload size varies from 2 to 30. Note that the random variables applied in our experiments are generated through Monte-Carlo simulation. We randomly designated the inputting order of the workloads assigned to a specific chip, and it keeps the same in both the baseline case and the cases with the proposed on-line techniques for the fair comparison. In the baseline case, each workload blindly selects a region of the chip for its execution, and we also name it random resource allocation. Four representative synthetic traffic patterns are used to simulate the traffic, they are uniform random (URT), transpose (TRA), bit-complement (BC), and tornado (Tor), which have been widely used in NoC design evaluation by prior studies [5, 9, 20].

We simulated the traffic pattern in each workload, the sources and destinations of the packets are limited to a region allocated to the workload. We modified the Garnet simulator to periodically inject packets into routers covered by the workloads during a period of 1 million cycles (including 100K warm-up cycles). Both one-flit and five-flit long packets are injected. Note that when a core is allocated to more than one application, its router will take the responsibility to send/receive packets from applications using the core; meanwhile, the total number of packets injected into the router will be the sum of packets injected by each application. As explained in Section 3, the region allocated to an application would be rotated when our techniques are enabled. Since a packet's destination is usually computed based on it source location when simulating the traffic pattern, the region-based rotation could cause a different destination for the packets starting from the source. In order to simulate exactly the same traffic when the proposed techniques are applied, we mimic the traffic by recording each packet's injection time and its source and destination regarding to each application in the baseline case.

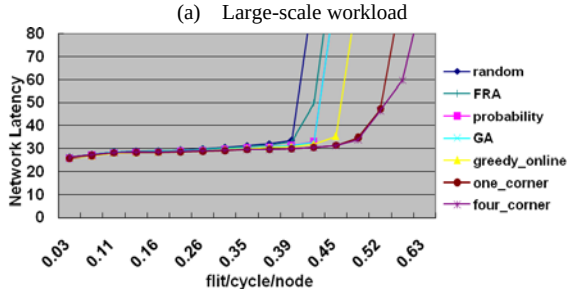
V. Experimental Results And Evaluation

In this section, we evaluate the effectiveness of the proposed techniques in reducing the network latency

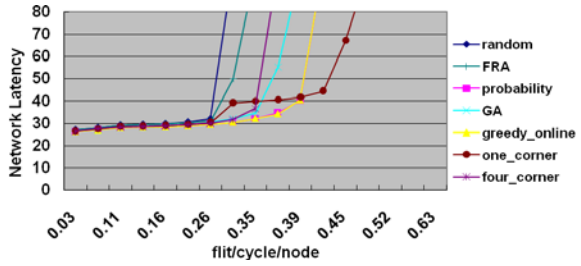
when dimension-ordered routing and adaptive routing algorithms are applied respectively. In order to improve the performance, one can just consider the maximum free region and simply assign the workload into that area without moving or rotating workloads. (We name this methodology as free-region aware algorithm or FRA in short.) In this study, we compare our techniques with both random and FRA methods.



(a) Large-scale workload



(b) Small-scale workload



(c) The Mix of large-scale and small-scale workload

Figure 11. The effectiveness of off-line and on-line algorithms for different workload based group using Uniform Random Traffic and dimension-ordered routing algorithm.

As described in Section IV, the simulated workloads have various sizes. A group of workloads being assigned to the multi/many-core processor can be composed of (1) all large-scale workloads which request a large number of cores (e.g. more than 10 cores), (2) all small-scale workloads which request a small number of cores (e.g. less than 10 cores), and (3) both large-scale and small-scale workloads. In another word, the groups of workloads can be classified into three different types: large-scale workload based (L), small-scale workload based (S), and the mix of large-scale and small-scale workload based (M). Fig. 11 show the network latency generated by off-line (probability-based searching algorithm and genetic algorithm), online algorithms (probability-based searching, four-corner, and one-corner based algorithms), free-region aware algorithm (FRA), and the baseline case (random resource allocation) for the three types of workload group when the uniform random traffic pattern and dimension-ordered routing algorithm are used. As Fig. 11 shows, our proposed technique exhibit strong capability in performance improvement across different workload groups. On average, our five algorithms reduce the network latency by 31% and 22% compared to the baseline case and FRA, respectively.

One may notice that the NoC gets saturated with lower packet injection rate in large-scale workload based groups compared to the other two types of groups. Because large-scale workload introduces communications between nodes with longer distance, packets experience a larger number of routers than those in small-scale workloads, which results in longer network latency. In addition, the workloads overlapping are difficult to avoid among large-scale workloads given the NoC resources.

As Fig. 11 shows, our offline algorithm (i.e. probability-based algorithm) reduces network latency 8% compared to the online algorithms in large-scale workload based groups, because it performs comprehensive search to obtain the optimum resource allocation, the benefit of exhausted search becomes more obvious when it is difficult for the online algorithms to achieve the best solution in avoiding the overlapping among large-scale workloads. Interestingly, the online algorithm outperforms the other offline algorithm, GA, in large-scale workload based groups. Because the effectiveness of GA highly depends on the initial layout of the workloads. On the other hand, our online algorithm (i.e. four-corner algorithm) outperforms other algorithms by 16% in small-scale workload based groups. The fundamental idea of four-corner algorithm is to push the workloads into corners to reduce the possibility of workloads overlapping, and it achieves better performance when the size of workloads is small. However, one-corner algorithm outperforms four-corner algorithm in the mix workload based group. Because in the mix workload group, the four-corner algorithm aggressively pushes the large-scale workloads into the corner, it would result in high fragmentation and reduce the size of the maximum free region in the NoC for future inserted workloads.

Fig. 12 shows the effectiveness of our offline and online algorithms when dimensional-order routing (DOR) and adaptive routing algorithm (e.g. regional-congestion aware) are applied with the four representative traffic patterns (URT, TRA, BC, and Tor). We used DOR_URT, DOR_BC, DOR_TRA, DOR_Tor, RCA_URT, RCA _BC, RCA_TRA and RCA_Tor describe the combination of routing algorithms and traffic patterns. As can be seen, our techniques are applicable to different routing algorithms and traffic patterns. On average, the online and offline algorithms reduce the network latency 27% and 18% compared to the baseline case and the free-region aware resource allocation.

VI. CONCLUSIONS

The use of congestion-aware packing helps constraining the coherency protocol within the domain boundaries thus preventing interference with traffic from

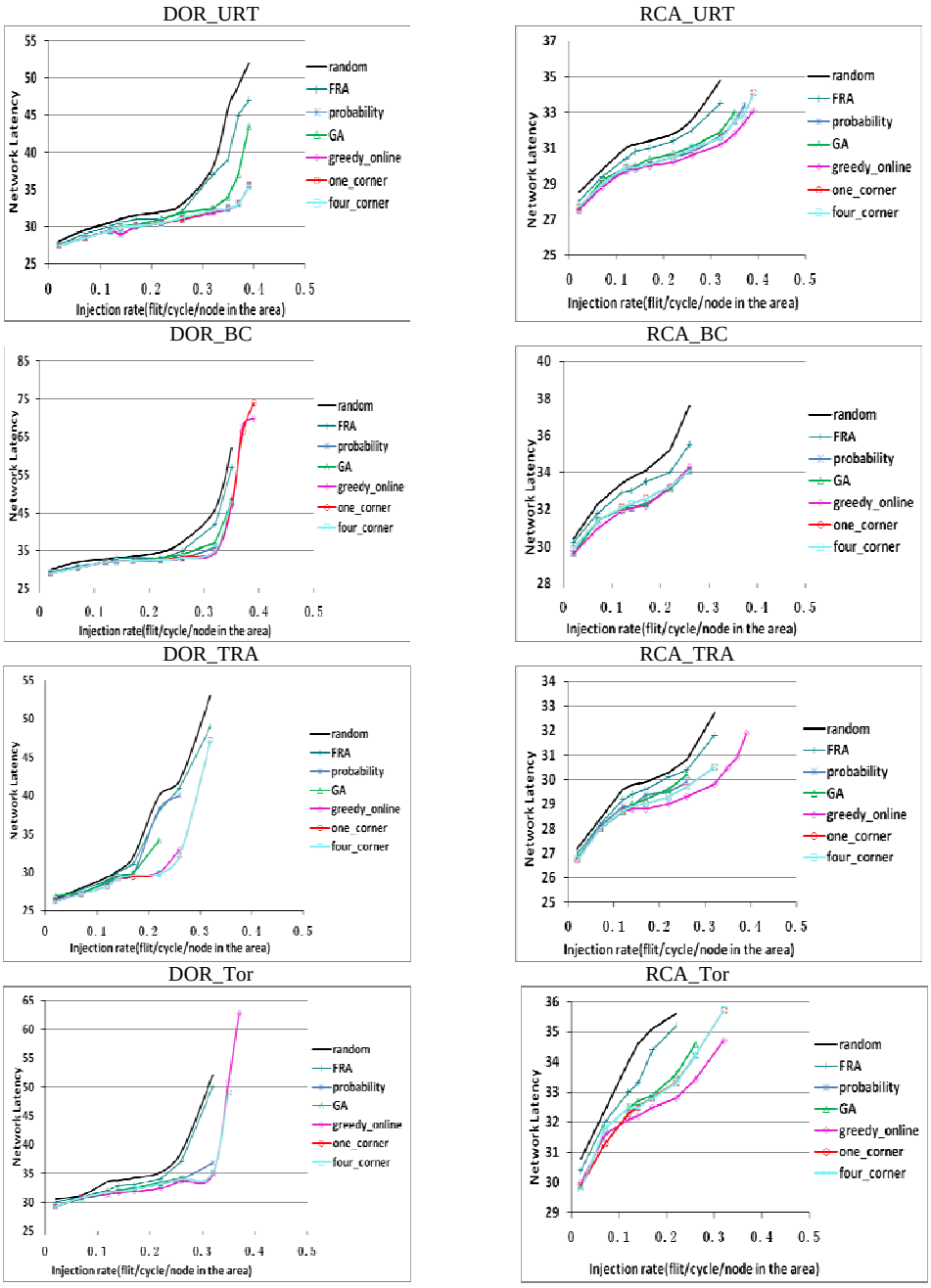


Figure 12. Simulation results for different injection

other regions of the chip. In addition, packing algorithms also benefit NoC partitioning. An application domain is defined as the complete set of cores used by the application. The definition of these domains helps techniques that ensure minimal fragmentation (both external and internal) when applications are mapped onto a group of cores. Our proposed spatial-based off-line an on-line congestion-aware NoC resource allocation techniques that can cost-effectively reduce NoC traffic interference among concurrently running applications and achieve the optimal NoC throughput at rut-time.

ACKNOWLEDGMENT

The work is supported in part by “the Fundamental Research Funds for the Central Universities”.

REFERENCES

[1] Intel, From a Few Cores to Many: A Terascale Computing Research Overview, 2006. http://download.intel.com/research/platform/teraacal/

[2] W. J. Dally, B. Towles, Route Packets, Not Wires: On-Chip Interconnection Networks, DAC, 2001.

[3] L. Benini and G. De Micheli, Networks on Chip: A New Paradigm for Systems on Chip Design, Proc. DATE,418-419, 2002.

[4] A. A. El Farag, H. M. El-Boghdadi, S. I. Shaheen, Improving Utilization of Reconfigurable Resources Using Two Dimensional Compaction, DATE, 2007.

[5] P. Gratz, B. Grot and S. W. Keckler, Regional Congestion Awareness for Load Balance in Networks on Chip, HPCA, 2008.
[6] Jongman Kim, Dongkook Park, T. Theocharides, N. Vijaykrishnan, C. R. Das, A Low Latency Router Supporting Adaptivity for On-Chip Interconnects, DAC, 2005.
[7] J. W. Lee, Man Cheuk Ng, Asanovic, K. Globally-Synchronized Frames for Guaranteed Quality-of-Service in On-Chip Networks, ISCA, 2008.
[8] U. Y. Ogras and R. Marculescu, Prediction-based Flow Control for Network-on-hip Traffic, DAC, 2006.
[9] E. Baydal, P. Lopez, J. Duato, A family of Mechanisms for Congestion Control in Wormhole Networks, IEEE Transactions on Parallel and Distributed Systems, 16(9), 772-784, 2005.
[10] J. Duato, I. Johnson, J. Flich, F. Naven, P. Garcia, T. Nachiondo, A New Scalable and Cost-effective Congestion Management Strategy for Lossless Multistage Interconnection Networks, HPCA, 2005.
[11] Jongman Kim, Dongkook Park, T. Theocharides, N. Vijaykrishnan, C. R. Das, A Low Latency Router Supporting Adaptively For on-chip Interconnects, DAC, 2005.
[12] R. Mullins, A. West, S. Moore, Low-latency Virtual-channel Routers for On-chip Networks, ISCA, 2004.
[13] Daeho Seo, Akif Ali, Won-Taek Lim, N. Rafique, Near-optimal Worst-case Throughput Routing for Two-dimensional Mesh Networks, ISCA, 2005.
[14] Arjun Singh, W. J. Dally, A. K. Gupta, B. Towles, GOAL: A Load-balanced Adaptive Routing Algorithm For Torus Networks, ISCA, 2003.
[15] M. Thottethodi, A. R. Lebeck, S. S. Mukherjee, Self-tuned Congestion Control for Multiprocessor Networks, HPCA, 2001.
[16] Chen-Ling Chou, U. Y. Ogras, R. Marculescu, Energy- and Performance-Aware Incremental Mapping for Networks on Chip With Multiple Voltage Levels, IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems, 27(10),1-14,2008.
[17] S. Murali, M. Coenen, A. Radulescu, K. Goossens, G. De Micheli, Mapping and Configuration Methods for Multi-use-case Networks on Chips, DAC, 2006.
[18] E. Carvalho, N. Calazans, F. Moraes, Heuristics for Dynamic Task Mapping in NoC-Based Heterogeneous MPSoCs, IEEE Workshop on Rapid System Prototyping, 2007.
[19] N. Agarwal, T. Krishna, L.-S. Peh and N. K. Jha, GARNET: A Detailed On-Chip Network Model inside a Full-System Simulator, ISPASS, 2009.
[20] V. Soteriou, Hangsheng Wang, L. Peh. A Statistical Traffic Model for On-Chip Interconnection Networks, pp. 104 - 116, MASCOTS, 2006.

Jingling Yuan, Dr. Jingling Yuan is an associate professor in Computer Science and Technology School at Wuhan University of Technology. She received a Ph.D. in Civil Engineering from Wuhan University of Technology. Her research interests include artificial intelligent, machine learning, multicore architecture analysis and resource allocation.

Xin Fu, Dr. Xin Fu is an assistant professor in the Department of Electrical and Computer Engineering at University of Kansas. She received a Ph.D. in Computer Engineering from University of Florida. Her research interests include computer architecture, hardware reliability, the impact of nano-scale technology scaling on multi-core processors, on-chip interconnection network.

Tao Li, Dr. Tao Li is an associate professor in the Department of Electrical and Computer Engineering at the University of Florida. He received a Ph.D. in Computer Engineering from the University of Texas at Austin. His research interests include computer architecture, virtualization, energy-efficient and dependable data center, cloud computing platforms, the impacts of emerging technologies and applications on computing, and evaluation of computer systems.

Minlong Yang, Mr Yang is a master student in Computer Science and Technology School at Wuhan University of Technology. His research interests include multicore architecture analysis and resource allocation.