2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing

Efficient Congestion-Aware Scheme for Wireless On-Chip Networks

Amin Rezaei
University of Louisiana at Lafayette, USA
(me@aminrezaei.com)

Masoud Daneshtalab
KTH Royal Institute of Technology, Sweden
(masdan@kth.se)

Maurizio Palesi
Kore University, Italy
(maurizio.palesi@unikore.it)

Danella Zhao
University of Louisiana at Lafayette, USA
(dzhao@cacs.louisiana.edu)

***Abstract*— Wireless NoC is becoming popular to be a promising future on-chip interconnection network as a result of high bandwidth, low latency and flexible topology configurations provided by this emerging technology. Nonetheless, congestion occurrence in wireless routers negatively affects the usability of high speed wireless links and considerably increases the network latency; therefore, in this paper, a congestion-aware platform (CAP-W) is introduced for wireless NoCs in order to reduce both internal and external congestions. The whole platform of CAP-W consists of an adaptive routing algorithm that balances utilization of wired and wireless networks, a dynamic task mapping approach that tries to minimize congestion probability, and a task migration strategy that considers dynamic variation of application behaviors. Simulation results show significant gain in congestion control over PEs of wireless NoC, compared to state-of-the-art works.**

Keywords—Network-on-Chip; Wireless Network-on-Chip; Congestion; Dynamic Application Mapping; Adaptive Routing Algorithm; Task Migration

I. INTRODUCTION

Based on the report of International Technology Roadmap for Semiconductors (ITRS) [1], Many-Core Systems-on-Chips (MCSoCs) may integrate hundreds of Processor Elements (PEs) connected based on a Network-on-chip (NoC) [2] communication infrastructure. NoC provides a regular platform for connecting system resources and makes the communication scalable and flexible compared to traditional bus or hierarchical bus architectures. Despite the fact that NoC-based architectures have a lot of advantages, their multi-hop nature has a negative impact on latency and power consumption especially as the network size increases; therefore, alternative technologies such as wireless NoC, 3D NoC, and photonic NoC were introduced [3-5].

Wireless NoC is becoming popular to be a promising future on-chip interconnection network as a result of high bandwidth, low latency and flexible topology configurations provided by this emerging technology. However, two major shortcomings of wireless NoC are extensive area and power overheads that wireless transceivers impose to the system. Thus, instead of a single NoC spanning the entire system, as is traditional, a hybrid wireless NoC is proposed using both wired and wireless links [6]. Furthermore, a Hierarchical Wireless NoC Architecture (HiWA) along with performance evaluation parameters is introduced where the network is divided into subnets [7]. Intra-subnet nodes communicate through wire links while inter-subnet communications are almost handled by single-hop wireless links. Also, a Wireless Router (i.e., a router equipped with a wireless interface, WR) placement is proposed for HiWA to allocate optimal number of WRs across the network [8].

NoC-based many-core systems face an extremely dynamic workload where applications, as sets of communicating tasks, map to the many-core system at run-time. The overall performance of the NoC is highly correlated to the network congestion [9]. Congestion not only increases the network latency severely [10] but also raises the network power consumption significantly [11]. Since each WR is shared by a cluster of processing elements, WRs are more vulnerable to congestion than Conventional Routers (CRs); therefore, in this paper, a Congestion-Aware Platform for Wireless NoC (CAP-W) is introduced to reduce system congestion. Experimental results show that CAP-W can reduce both internal and external congestions significantly. Furthermore, more than 80% of the packets are delivered by one-hop distance using CAP-W scheme. In addition, CAP-W also increases system utilization about 20% compared to the conventional NoC.

The rest of the paper is organized as follows. Section II reviews backgrounds and motivations. In section III an adaptive routing for HiWA is introduced. An efficient dynamic application mapping for HiWA is proposed in section IV. Section V presents a task migration strategy for HiWA while the results are discussed in section VI and finally, the conclusions are given in the last section.

II. BACKGROUNDS AND MOTIVATIONS

Recent growth in silicon integrated circuit technology has permitted the integration of tiny transceivers antennas on a single chip, which results in introducing wireless NoC. A low Terahertz (324GHz) frequency generator is realized in 90nm CMOS [12]. Moreover a signal source operating near 410GHz that is fabricated using low-leakage transistors in a 6M 45nm digital CMOS technology is reported [13]. Based on these techniques, the output power level of the on-chip millimeter-wave generator is expected to be as high as -1.4dBm in the 32nm CMOS process, which is large enough for on-chip short distance communication [14]. Following the rule of thumb in RF design, the maximum available bandwidth is 10% of the carrier frequency. According to this experimental estimation, up to 16 channels can be available for wireless NoC in the range of 100 to 500GHz. With recent developments of millimeter-wave circuits, bandwidths of hundred GHz can be reachable. In addition to the bandwidth, wireless NoC requires low-power on-chip wireless transceivers. Silicon Mach-Zehnder electro-optic modulator at data rates up to 10Gb/s with low RF power consumption of only 5pJ/bit [15] is commercially available.

Since each WR is shared by many PEs, an efficient task allocation technique is required to balance the utilization of WRs and reduce congestion. However, as task allocation is known as NP-hard problem, different heuristics dealing with dynamic management of workload in multi-core systems, such as Nearest Neighbor (NN) and Best Neighbor (BN) are presented [16], [17]. In these heuristics, a clustering mechanism for the *first node* selection is considered. A set of cluster nodes are assumed to select the *first node* of the mapping algorithm among them. In another approach called Incremental Approach (INC) [18], the mapping problem is break down into two steps: the region selection, and the task allocation. In the region selection step, it starts from the closest node to the Central Manager (CM) and

978-1-4673-8776-7/16 $31.00 © 2016 IEEE
DOI 10.1109/PDP.2016.88

include it in the region. Then, it iteratively adds nodes to the selected region trying to keep both the selected region and the remaining nodes contiguous. Afterward, in the task allocation step, application tasks are mapped inside the selected region. As an advanced approach, CoNA [19] selects the closest node to the CM with all its neighbors available. Thus a minimum number of available nodes are assured. Then, Task Graph (TG) is traversed in breath-first order and tasks are mapped onto the neighboring of their parents in which a smaller square is formed. Moreover, in [20] a Smart Hill Climbing (SHiC) algorithm is introduced. SHiC uses a square factor (SF) model to approximate the contiguous available nodes around a given node. However, previous papers, which have been mentioned above, proposed task mapping algorithms for conventional NoCs and they did not cover specified congestion-aware task mapping for wireless NoCs. In [21] a Dynamic Application Mapping Algorithm (DAMA) is presented and evaluated for wireless NoCs that is adapted in CAP-W as application mapping scheme.

Due to significant vibration of application behaviors, even optimal congestion-aware task mapping may not meet the best performance, which makes some re-mapping strategies take behavior variation into consideration. Task migration has been traditionally studied in distributed systems for dynamic load balancing. However, with the increasing popularity of MCSoCs in modern embedded systems, task migration has also gained research attention in this domain. By efficiently trace dynamic variation of workload specifications, task migration can improve overall performance of the system. In [22] a lightweight migration mechanism for bus-based MPSoC is presented. The task migration method relies on modification of program to define the checkpoints. When running to a checkpoint, the program checks whether there is a migration request for the current task. The authors in [23] proposed a methodology based on virtual channels to create connections that provide low latency and low power paths for the task migration flows. They adopted a 2D-mesh NoC, creating sub-meshes which may contain one or more cores. However, the proposed task migration strategies are for conventional NoCs and not considering wireless NoCs which our paper is targeted.

On top of the application mapping and migration schemes, adaptive routing algorithm can also alleviate the network congestion. Therefore, we also apply an adaptive routing algorithm in order to reduce congestion on WRs.

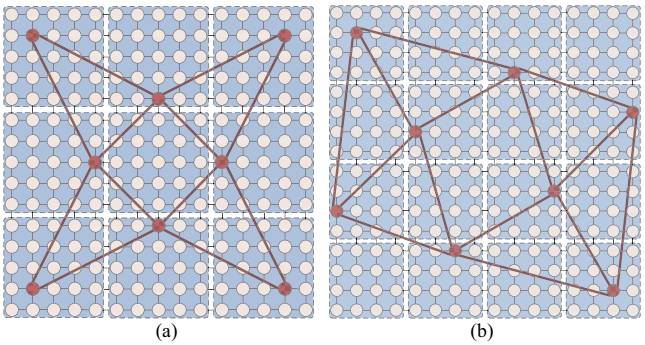


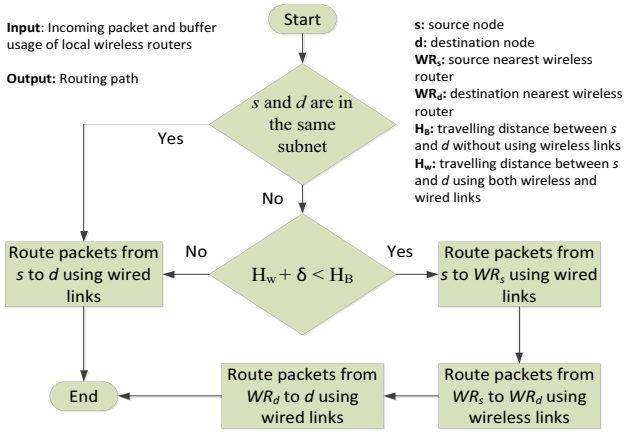
Fig. 1. Illustration of HiWA architecute (light-colored nodes are CRs and high-colored nodes are WRs) (a) 225-node HiWA (b) 256-node HiWA

The backbone of HiWA is based on 2D mesh NoC. Whenever it is necessary, one of the Conventional Routers (CRs) is replaced by a Wireless Router (WR). A WR can wirelessly communicate with the other WRs in other subnets. In addition, a WR can perform wired communications like a CR. Fig. 1a shows a 225-node HiWA which is divided into 9 subnets of 5×5 nodes. Fig. 1b shows a 256-node HiWA which is divided into 16 subnets of 4×4 nodes. Since our focus is on congestion control, finding the optimal subnet size and the placement of WRs [8] are out of the scope of this paper.

III. Adaptive Routing Algorithm for CAP-W

In HiWA, the communication is handled by wired, wireless paths or a combination of wired and wireless paths. This can be seen as a hybrid network that has been characterized by adding express paths (i.e. wireless links) to a 2D mesh NoC; therefore, whether the packet will take or not take the express paths is an important decision to make. One of the benefits of partitioning is that intra-subnet communications are handled through wire paths while for inter-subnets communications a function of hop counts and congestion is used in order to select the efficient path.

FLOWCHART I. CAP-W ROUTING APPROACH



Flowchart I represents CAP-W routing flowchart. Since each WR is shared by several nodes, there is a possibility of congestion over WRs. In order to balance the utilization of wired and wireless interconnections, a balance parameter called δ is added to the routing decision. The value of δ depends on the network size and the utilization of wireless interconnection.

$$\delta = C \times u \quad (1)$$

As it is shown in Equation 1, δ consists of two major parameters, the static parameter (C) defined as the ratio of WRs to CRs and a dynamic parameter (u) that exponentially increases by wireless link utilization. In general, the larger the network size or the higher the link utilization is, the larger the δ. In each router, there is a table stores and updates the δ value based on different situations. Once δ increases, lower priority will be given to wireless links which can help alleviating the congestion in the wireless network. The Equation 1 is based on the WRs utilization. Other metrics such as power consumption and thermal analysis can also be taken into consideration. One of the principal subjects should be addressed in networks using wormhole switching is the deadlock avoidance. Although using the Dimension Order Routing (DOR) in each of wired and wireless networks guarantees deadlock freedom, when packets

transmit through both wired and wireless paths, there is a possibility of channel dependency as shown in Fig. 2a. In order to overcome the problem, virtual channels are taken into account. In each input port of the routers two sets of virtual channels are used (Fig. 2b). One of them is for traffic transmission using nearest wireless router while the other one is utilized for either the wired network or traffic transmission of the wireless router to the destination node.

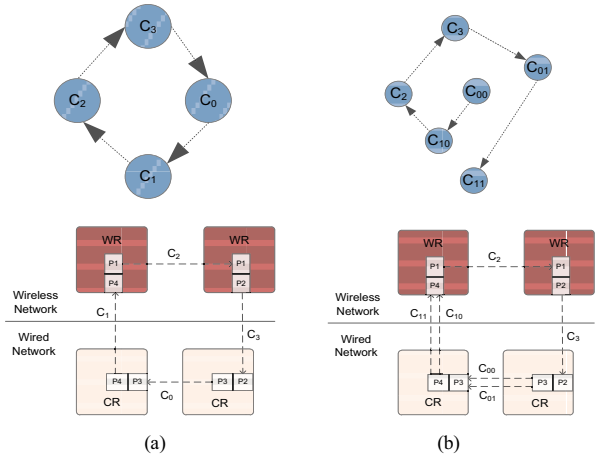


Fig. 2. A deadlock prone situation in CAP-W routing approach (a) Without using virtual channels (b) By using virtual channels

IV. Dynamic Task Mapping Approach for CAP-W

Congestion has a negative effect on the network performance and greatly increases the network latency. There are many works have tried to reduce the network congestion in different aspects, like routing as we discussed in section III. In the absence of an efficient mapping approach, the utilization of WRs will increase considerably where adaptive routing cannot fully prevent network performance degradation. In high traffic workloads adaptive routing may not use WRs, as δ increases. This leads to increase the network latency. Thus, an efficient application mapping approach is essential in wireless NoCs to reduce congestion over WRs.

Two types of congestion can be considered from dynamic application mapping perspective: external and internal congestions. External congestion happens when a network channel is contented by edges of different applications. To decrease external congestion probability, the application mapped region should be as compact as possible and minimally fragmented. On the other hand, the internal congestion happens when a network channel is contended by edges of the same application. A directed graph, named as Task Graph (TG), represents each application in the system. Each vertex represents one task of the application, while each edge stands for a communication between the source task and the destination task as shown in Equation 2. TG of an application with 6 tasks is shown in Fig. 3. The amount of data transferred from the source task to the destination task is written on the edge.

$$\forall t_i \in T, \forall e_{i,j} \in E, A_p = TG(T, E) \qquad (2)$$

CAP-W task mapping approach consists of three steps [21]. The first step is to select the *first node* to map. The second step is picking up the *first task* of the application with the largest number of edges to be mapped onto the *first node*, which reduces internal congestion probability. After all, in order to reduce the external congestion establishing a contiguous area of available nodes around the *first node* to map the rest of the tasks of the application is taking into account. In the following we present each step separately.

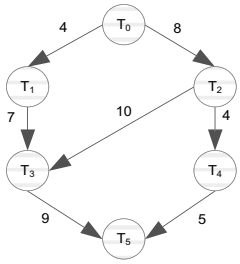


Fig. 3. Task graph of an application with 6 tasks and 7 edges

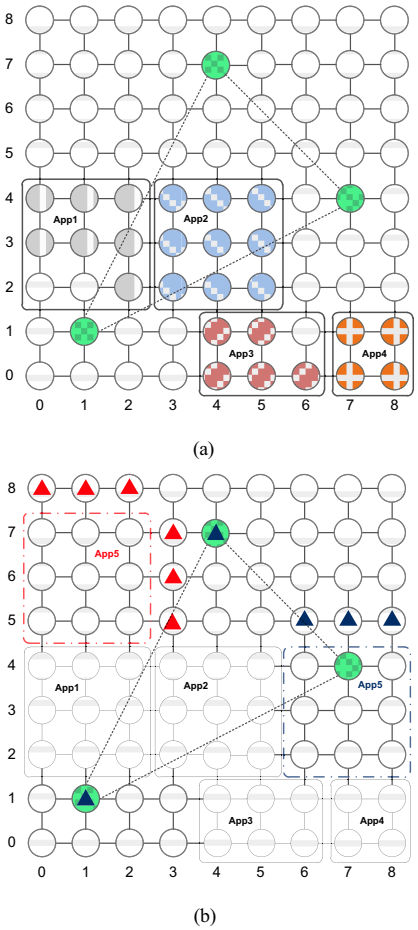


Fig. 4. CAP-W square factor calculation

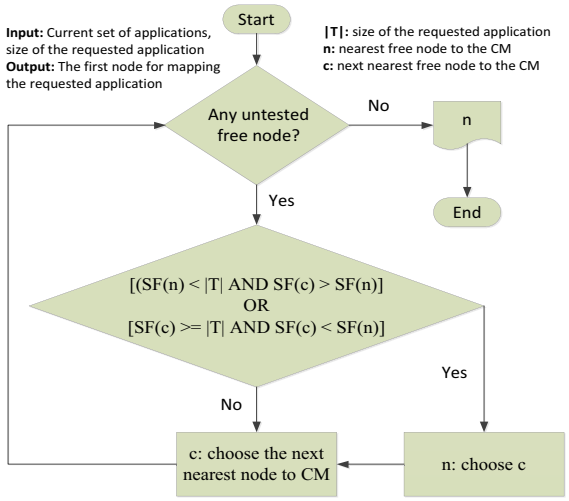
A. First Node Selection

The most contiguous area is almost circular [18]. However, because adjacent regions share network links, choosing a circular region for an application in the mesh network increases the external congestion. As an alternative, when tasks are mapped onto a rectangular region of a network with minimal routing, all packets will be routed inside the region border and there will be no external congestion. The most contiguous rectangle is the

square, and thus it is preferred in CAP-W. The Square Factor (SF) of a node is the estimated number of contiguous, almost square-shaped, available nodes around the *first node*. Accordingly, the suitable *first node* for mapping of an application would be the node with the SF equal to the application size [20].

Each running application in the system is modeled as a rectangle characterized by its corner nodes. Regarding the rectangle model of a running application, there might be some nodes within the rectangle which do not belong to the application. In this work, the rectangle of each application is modeled in such a way that minimizing the number of these nodes while trying to keep the model almost in the square-shaped. The rectangle models of four running applications are shown in Fig. 4a. For instance, the rectangle of the application 1 has two nodes which do not belong to it (i.e. $n_{1,2}$ and $n_{0,2}$). Also the rectangle of the application 3 includes one node (i.e. $n_{6,1}$) which is not a part of the application. However, they are the best fitting rectangles in order to stay close to square-shaped.

FLOWCHART II. CAP-W *FIRST NODE* SELECTION



To calculate the SF for each node:

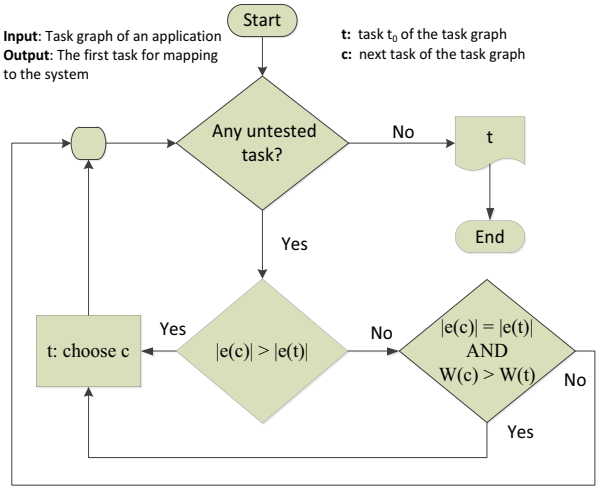
- First, the largest square centered on the node is found, where it fits within the mesh limits and has no overlap with other running applications of the system. This is shown in Fig. 4b for the node $n_{7,3}$ which is the *first node* of the application 5.
- Second, there might be also some more nodes beyond the square borders not belonging to system rectangles, as marked with triangle in Fig. 4b. These nodes have one-hop distance to one of the nodes within the square border. They are counted in order to prevent available nodes from being isolated while keeping the mapped area close to a square shape.
- Finally, The SF of a given node is calculated by adding the nodes in the area of the largest square, with the available nodes beyond the square borders. For instance, the SF for $n_{7,3}$ will be the square area nodes, 9, summed up with marked nodes, 5, which is 14. As shown in Fig. 4b, two WRs (i.e. $n_{4,7}$ and $n_{1,1}$) are also counted in SF factor of $n_{7,3}$ because they have one-hop distance to the node $n_{7,4}$ that is inside the square border of the application.

The *first node* selection algorithm of CAP-W starts from the nearest free node to CM and walks through the network to find the appropriate *first node*. It first looks for the node with the smallest SF value which is larger than or equal to the application size. Otherwise, the node with the largest SF value is preferred. Note that, when there are two nodes with equal SF, the closest one to CM is preferred to decrease the incurred defragmentation of remaining nodes. Also, in order to reduce congestion over WRs, they are not chosen as *first node* of the application by CAP-W *first node* selection. The two candidates for the *first node* of the application 5 are shown in Fig. 4b. The CAP-W *first node* selection will choose the node $n_{7,3}$ because the SF factor of this node is 14 which is smaller than the node $n_{1,6}$ with SF factor of 15. Existing WRs, which have express paths to other WRs, will help the application to be mapped as contiguous as possible. In fact, WRs play the role of spreading contiguity across the whole system. Flowchart II shows CAP-W *first node* selection.

B. First Task to Map

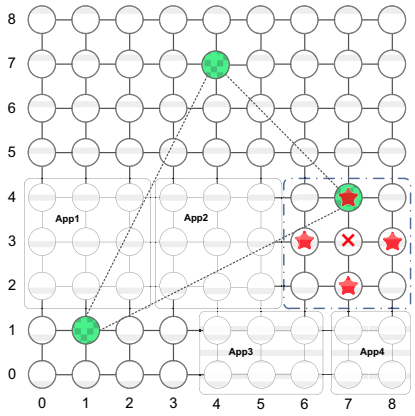
The task with the largest number of edges is selected to be mapped onto the *first node*. This provides the largest possible number of available nodes around the *first task*; therefore, the edges of the *first task* can mostly be controlled by one-hop links, which reduces the congestion probability for the *first task*. If there is more than one task having the largest number of edges, then the *first task* would be the one with the most intensive communication. For example, in Fig. 3, both tasks t_2 and t_3 have 3 edges. Accordingly, since the total communication weight of t_3 is more than that of t_2 (i.e. 26 vs. 23), t_3 is selected as the *first task* to be mapped. Flowchart III demonstrates CAP-W *first task* selection.

FLOWCHART III. CAP-W *FIRST TASK* SELECTION

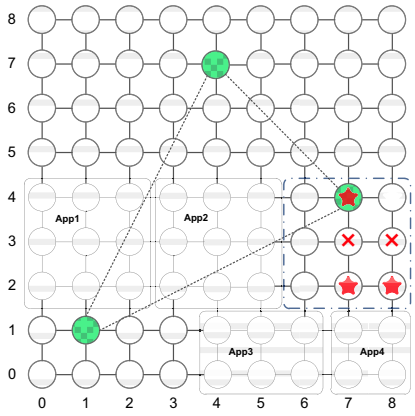


C. Neighborhood Allocation

After the *first task* is mapped onto the *first node*, the task mapping approach of CAP-W assumes the TG to be undirected and traverses tasks through their predecessor tasks in the breadth-first order, starting from the *first task*. Considering the set of available nodes in the closest neighborhood of the predecessor task, tasks are mapped onto the nodes which fit into the smallest square with the *first node*.



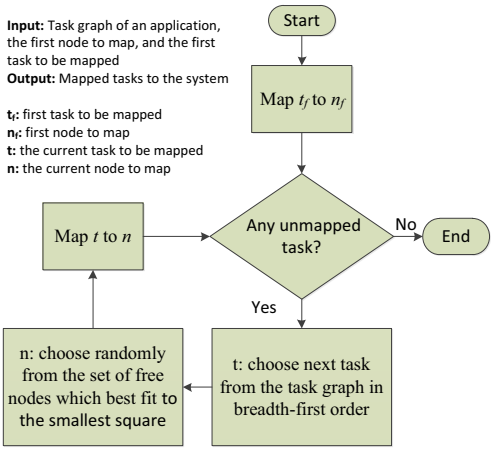
(a)



(b)

Fig. 5. CAP-W neighborhood allocation

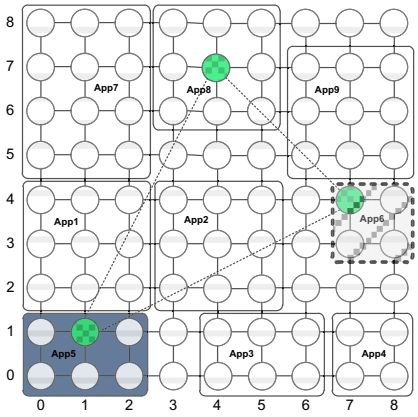
FLOWCHART IV. CAP-W TASK MAPPING APPROACH



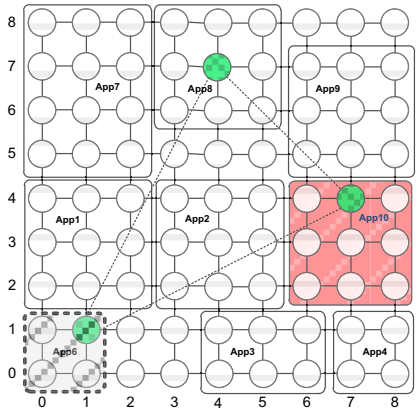
For example, considering the application 5 in Fig. 4b, after mapping the *first task* to the *first node* (i.e. $n_{7,3}$) the second node will be randomly chosen from one of the nodes of the set A={$n_{6,3}$,$n_{7,2}$, $n_{7,4}$,$n_{8,3}$}. Then supposing that $n_{8,3}$ is chosen from A, the new set for choosing the third node will be B={$n_{7,2}$, $n_{8,2}$, $n_{7,4}$, $n_{8,4}$}. The demonstration is illustrated in Fig. 5. As a result, CAP-W task mapping approach maps the communicating tasks onto the closest neighborhood, while keeping the mapped area as close to square as possible. CAP-W task mapping approach is shown in Flowchart IV.

V. TASK MIGRATION STRATEGY FOR CAP-W

Due to dynamic variation of application behavior, even optimal congestion-aware mapping may not meet the best performance. Thus, a re-mapping strategy is required in order to consider such variations. Task migration is a dynamic task re-mapping mechanism, which traces dynamic variation of workload behavior. Task migration in a many-core system is defined as transferring a task from the core where it is currently running to another core and then resuming its execution there in such a way that some of system performance objectives are improved.



(a)



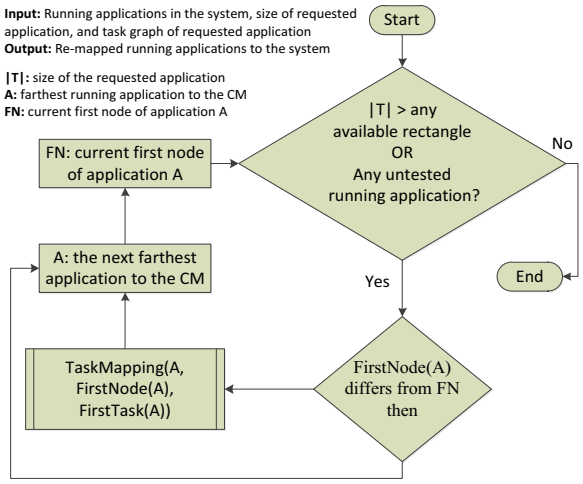
(b)

Fig. 6. CAP-W task migration

As depicted in Fig. 6a, suppose 9 applications are running in the system. After a while, application 5 finishes the execution and application 10 with 8 tasks requests to enter the system. Although there are some idle core the new application can be assigned to, assigning tasks of a single application in non-adjacent cores is not aligned with the contiguous mapping approach. In this condition, mapping application 10 onto the system increases external congestion probability. To solve this problem, CAP-W exploits an agile task migration. As shown in Fig. 6b, migrating application 6 in the previous place of

application 5 provides enough contiguous region for application 10. Flowchart V shows CAP-W task migration scheme which starts only if there is not enough available cores for the incoming application to be fitted into a rectangle. It checks running applications starting from the farthest application to CM to find a better mapping for them. In the case of finding a better mapping area, the current application is transferred and dynamically remapped to the new place using the shortest path with the highest priority between the current and destination cores of each task. If the source and destination are not close to each other, the task migration usually uses wireless paths in order to expedite the task migration process. The task migration algorithm continues until enough contiguous cores are found for the requested application or as long as all the running applications are tested.

FLOWCHART V. CAP-W TASK MIGRATION SCHEME



VI. EXPERIMENTAL RESULTS

In this section, we assess the impact of CAP-W platform on improving the mapping results. Several set of applications with 4 to 35 tasks are generated using TGG [24] where the amount of data transferred from the source task to the destination task are randomly distributed between 4 to 16 flits of data. Also to measure the effectiveness of the CAP-W routing algorithm, some application benchmark suites selected from SPLASH-2 [25] are used. Experiments are performed using *XMulator* [26] an integrated simulation platform for interconnection networks. Different mapping and *first node* selection methods are evaluated over the network size varying from 8×8 to 16×16 nodes. A random sequence of applications is entered into the scheduler FIFO according to the desired rate, λ. The sequence is kept fixed in all experiments for the sake of fair comparison. Applications are scheduled based on the First Come First Serve (FCFS) policy and the maximum possible scheduling rate is called λ_{full}. An allocation request for the scheduled application is sent to the CM of the platform residing in the node $n_{0,0}$. In order to have a holistic view of the results and enable real case comparisons, each set of experiments are performed over ten million cycles where hundreds of applications enter and leave the system.

A. Evaluation Metrics

AMD: Cost of a packet delivery is related to the number of hops it traverses. Hence, a metric to evaluate a mapping is the Average Manhattan Distance (AMD) between tasks of the mapped application. Since the communicating nodes are placed close to each other, the smaller the value of AMD is, the lower the average packet latency.

$$AMD_{map(A)} = \frac{\sum_{\forall e_{i,j} \in E} MD(map(t_i), map(t_j))}{|E|} \quad (3)$$

AWMD: The packet delivery cost depends not only on the length of its path, but also on the size of the packet. Thus, a more precise evaluation is to also include the weight of edges. Average Weighted Manhattan Distance (AWMD) is the sum product of Manhattan Distance (MD) and all edges' weight of the mapped application, averaged by the total communication weights.

$$AWMD_{map(A)} = \frac{\sum_{\forall e_{i,j} \in E} W_{i,j} \times MD(map(t_i), map(t_j))}{\sum W_{i,j}} \quad (4)$$

MRD: To assess how contiguous the mapped region of an application is, Mapped Region Dispersion (MRD) factor is defined that is the mean value of all possible node pairs MD in the mapped region:

$$MRD_{map(A)} = \frac{\sum_{\forall t_i, t_j \in T} MD(map(t_i), map(t_j))}{\left(\frac{|T|}{2}\right)} \quad (5)$$

NMRD: To decrease external congestion probability, the application mapped region should be as compact as possible and minimally fragmented. As it is mentioned before, the most contiguous area, which has also the smallest MRD, is almost circular. Regarding the mesh topology of the network, however, a circular region will generate irregularity in remaining available nodes and more area fragmentation in long term. On the other hand, a rectangular allocation forms regular regions, decreases applications overlap and thus isolates their communications. Thus, the best mapped area would be square as it is the rectangle with the smallest MRD. It can be shown that the MRD of a square with |T| nodes will be:

$$MRD_{SQ(|T|)} = \frac{2 \times \sqrt{|T|}}{3} \quad (6)$$

Therefore, the Normalized Mapped Region Dispersion (NMRD) metric is defined which assesses the squareness of the mapped region independent of the size of the application. NMRD increases as the mapped area is getting more fragmented and less similar to a square shape:

$$NMRD_{map(A)} = 1 + \frac{\left|MRD_{map(A)} - MRD_{SQ(|T|)}\right|}{\left|MRD_{SQ(|T|)}\right|} \quad (7)$$

On the other hand, the internal congestion occurs when a network channel is contended by edges of the same application. Internal Congestion Ratio (ICR) is the number of edges of an application using the same communication channel (according to the XY algorithm) with respect to its total number of edges (|E|). Of note, we do not count overlapped edges that are originated from the same source. In such case, their injection is limited by source injection rate limit and they will never contend.

B. Latency Evaluation

The packet latency and values of different evaluation are summarized in Table I. Evaluation metrics are normalized to the

CAP-W results to ease comparison. The mapping results for applications with different packet sizes are the same, because the application TG and system behavior remain the same. The application injection rate is 2/3 λ_{full}. As can be seen, CAP-W outperforms the BN algorithm by 40% and 20% reduction in external and internal congestion factors, respectively. It, also, obtains 50% gain over the INC in internal congestion and more than 15% in external congestion. In addition, CAP-W achieves 25% gain in average for all evaluation aspects in comparison with the NN algorithm.

TABLE I. LATENCY EVALUATION FOR DIFFERENT MAPPING ALGORITHMS

	NN [16]	BN [17]	INC [18]	CAP-W
AMD	1.25	1.30	1.42	1.00
AWMD	1.29	1.32	1.36	1.00
NMRD	1.24	1.62	1.21	1.00
ICR	1.52	1.21	2.01	1.00

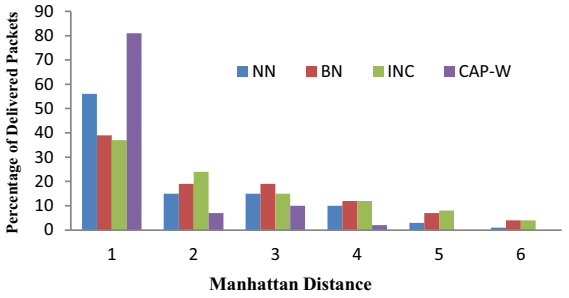


Fig. 7. Percentage of delivered packets in different path lengths

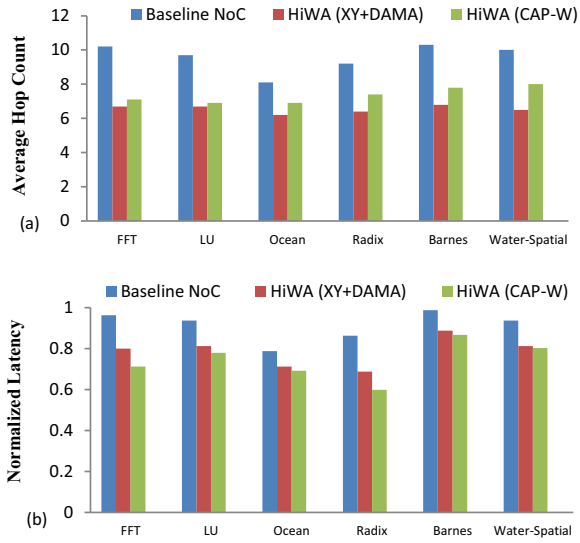


Fig. 8. (a) Hop count comparison (b) Latency comparision

As shown in [18], decreasing MD between tasks of application edges is an effective way to minimize the communication energy consumption of the application. We illustrate the percentage of packets that are delivered over different path lengths (i.e. MD). The experiments have been run for different algorithms in the injection rate of 2/3λ_{full}. As depicted in Fig. 7, more than 80% of the packets are delivered by one-hop distance using CAP-W scheme.

In addition, Fig. 8 shows the average hop counts and the average packet latency among the baseline NoC without any WR integration, HiWA using the XY routing and DAMA mapping, and HiWA equipped with CAP-W. As it can be seen, although HiWA with XY and DAMA outperforms the HiWA using CAP-W in average hop counts, it suffers from network latency caused by congestion in WRs.

C. Time Complexity Evaluation

The average number of clock cycles that is elapsed in CM to map applications with number of tasks varying from 4 to 10 is presented in Fig. 9a. The injected rate is set to 0.75 λ_{full}. Furthermore, the time complexity of different mapping for 8-task applications is presented in Fig. 9b, when the injection rate varies from 0.4 λ_{full} to λ_{full}. As can be seen, CAP-W provides a reasonable time complexity next to NN. As it is shown in Fig. 9b, all mapping algorithms scale well when the injection rate is increased.

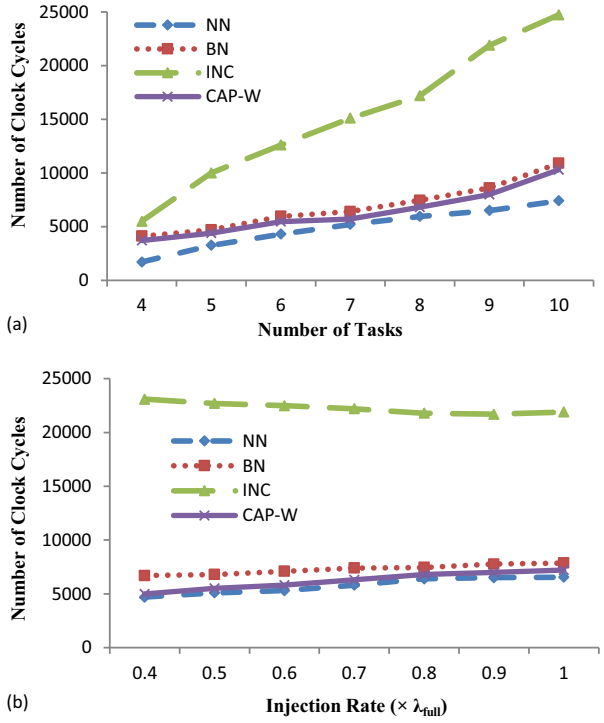


Fig. 9. Time complexity of mapping algorithms (a) over application sizes (b) over different injection rates

Furthermore, Table II shows the average number of clock cycles required for all the task migrations taken place in each injection rate. From Fig. 9b and Table II, the overhead time of task migration is negligible compared to mapping time complexity (i.e. about 0.1x in the worst case scenario).

D. System Utilization Measurement

System utilization is another important factor has been analyzed among the baseline NoC, HiWA with XY and DAMA, and HiWA with CAP-W. As shown in Fig. 10, HiWA equipped

with CAP-W increases the average system utilization compared to the baseline NoC and HiWA with XY and DAMA. In addition, the maximum system utilization is defined as the highest percentage of the utilization during the simulation time shown in Fig. 10. HiWA equipped with CAP-W also increases the maximum utilization compared to the others. As can be noticed, the proposed platform cannot reach 100% utilization because area fragmentation usually occurs due to the dynamic mapping policy (i.e. when applications do not exactly fit onto the many-core system).

TABLE II. TIME COMPLEXITY OF TASK MIGRATION

λ ($\times\lambda_{full}$)	0.4	0.5	0.6	0.7	0.8	0.9	1
Number of Clock Cycles	30	132	383	536	680	840	1080

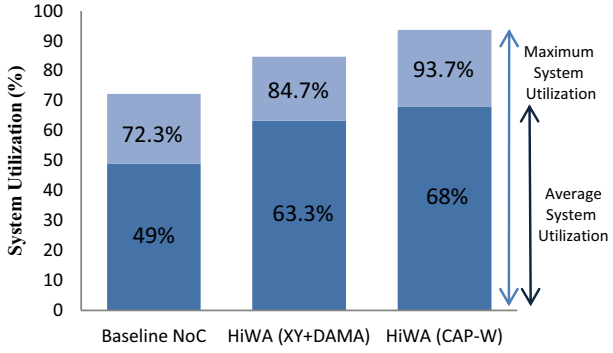


Fig. 10. System utilization

VII. CONCLUSION

In this paper, we proposed an efficient congestion-aware platform, CAP-W, for wireless NoC. CAP-W targets at reducing internal and external congestions, and consists of three main parts. First an adaptive routing algorithm that balances utilization of wired and wireless networks, second a dynamic task mapping approach that tries to minimize congestion probability, and third a task migration strategy that considers dynamic variation of application behaviors. Existing WRs, which have express paths to other WRs, help the system area to stay as contiguous as possible. In fact, WRs play the role of spreading contiguity across the whole system. Experimental results showed that CAP-W accomplish a reduced internal and external congestions as targeted. Also, more than 80% of the packets are delivered by one-hop distance using CAP-W scheme.

REFERENCES

[1] ITRS. International Technology Roadmap for Semiconductors, 2011 edition.

[2] L. Benini and G. D. Micheli, "Networks on chips: a new SoC paradigm," In *IEEE Computer,* Vol. 35, Issue 1, pp. 70-78, 2002.

[3] M. F. Chang, J. Cong, A. Kaplan, M. Naik, G. Reinman, E. Socher, and S. W. Tam, "CMP network-on-chip overlaid with multi-band RF-interconnect," In *Proceedings of IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 191-202, 2008.

[4] V. F. Pavlidis and E. G. Friedman, "3-D topologies for networks-on-chip," In *IEEE Transactions on Very Large Scale Integration*, Vol. 15, Issue 10, pp. 1081-1090, 2007.

[5] A. Shacham, K. Bergman, S. Member, and L. P. Carloni, "Photonic networks-on-chip for future generations of chip multiprocessors," In *IEEE Transactions on Computers*, Vol. 57, Issue 9, pp. 1246-1260, 2008.

[6] A. Ganguly, K. Chang, S. Deb, P. P. Pande, B. Belzer, and C. Teuscher, "Scalable hybrid wireless network-on-chip architectures for multicore systems," In *IEEE Transactions on Computers*, Vol. 60, Issue 10, pp. 1485-1502, 2011.

[7] A. Rezaei, F. Safaei, M. Daneshtalab, and H. Tenhunen, "HiWA: A hierarchical wireless network-on-chip architecture," In *Proceedings of IEEE International High Performance Computing & Simulation (HPCS)*, pp. 499-505, 2014.

[8] A. Rezaei, M. Daneshtalab, F. Safaei, and D. Zhao, "Hierarchical approach for hybrid wireless network-on-chip in many-core era," In *Elsevier International Journal of Computers and Electrical Engineering*, 2015. (DOI: 10.1016/j.compeleceng.2015.10.007)

[9] C. L. Chou and R. Marculescu, "Contention-aware application mapping for network-on-chip communication architectures," In *IEEE International Conference on Computer Design (ICCD)*, pp. 164-169, 2008.

[10] J. W. Brand, C. Ciordas, K. Goossens, and T. Basten, "Congestion-controlled best-effort communication for networks-on-chip," In *Proceedings of Design, Automation and Test in Europe (DATE)*, pp. 1-6, 2007.

[11] S. Ma, N. E. Jerger, and Z. Wang, "DBAR: an efficient routing algorithm to support multiple concurrent applications in networks-on-chip," In *Proceedings of International Symposium on Computer Architecture (ISCA)*, pp. 413–424, 2011.

[12] D. Huang, T. LaRocca, M. C. Chang, L. Samoska, A. Fung, R. Campbell, and M. Andrews, "Terahertz CMOS frequency generator using linear superposition technique," In *IEEE Journal of Solid-State Circuits*, Vol. 43, Issue 12, pp. 2730-2738, 2008.

[13] E. Seok, C. Cao, D. Shim, D. J. Arenas, D. B. Tanner, C. Hung, and K. K. O, "A 410GHz CMOS push-push oscillator with an on-chip patch antenna," In *IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 472-629, 2008.

[14] S. B. Lee, S. W. Tam, I. Pefkianakis, S. Lu, M. F. Chang, C. Guo, G. Reinman, C. Peng, M. Naik, L. Zhang, and J. Cong, "A scalable micro wireless interconnect structure for CMPs," In *Proceedings of the International Conference on Mobile Computing and Networking (MobiCom)*, pp. 217-228, 2009.

[15] W. Green, M. Rooks, L. Sekaric, and Y. Vlasov, "Ultra-compact, low RF power, 10 Gb/s silicon Mach-Zehnder modulator," In *Optics Express*, Vol. 15, Issue 25, pp. 17106-17113, 2007.

[16] E. L. Carvalho, N. L. V. Calazans, and F. G. Moraes, "Heuristics for dynamic task mapping in NoC-based heterogeneous MPSoCs," In *IEEE/IFIP International Workshop on Rapid System Prototyping (RSP)*, pp. 34-40, 2007.

[17] E. L. Carvalho, N. L. V. Calazans, and F. G. Moraes, "Dynamic task mapping for MPSoCs," In *IEEE Design & Test of Computers*, Vol. 27, Issue 5, pp. 26-35, 2010.

[18] C. L. Chou, U. Y. Ogras, and R. Marculescu, "Energy- and performance-aware incremental mapping for networks on chip with multiple voltage levels," In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 27, Issue 10, pp. 1866-1879, 2008.

[19] M. Fattah, M. Ramirez, M. Daneshtalab, P. Liljeberg, and J. Plosila, "CoNA: dynamic application mapping for congestion reduction in many-core systems," In *IEEE International Conference on Computer Design(ICCD)*, pp. 364-370, 2012.

[20] M. Fattah, M. Daneshtalab, P. Liljeberg, and J. Plosila "Smart hill climbing for agile dynamic mapping in manycore systems," In *ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1-6, 2013.

[21] A. Rezaei, M. Daneshtalab, D. Zhao, F. Safaei, X. Wang, and M. Ebrahimi, "Dynamic application mapping algorithm for wireless network-on-chip," In *Proceedings of IEEE Euromicro Conference on Parallel, Distributed and Network-Based Computing (PDP)*, pp. 421-424, 2015.

[22] S. Bertozzi, A. Acquaviva, D. Bertozzi, and A. Poggiali, "Supporting task migration in multi-processor systems-on-chip: a feasibility study," In *Proceedings of Design, Automation and Test in Europe (DATE)*, pp. 1-6, 2006.

[23] B. Goodarzi and H. Sarbazi-Azad, "Task migration in mesh NoCs over virtual point-to-point connections," In *Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pp. 463-469, 2011.

[24] "Task graph generator (TGG)." [Online]. Available: http://sourceforge.net/projects/taskgraphgen/.

[25] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 programs: characterization and methodological considerations," In *International Symposium on Computer Architecture (ISCA)*, pp. 24-36, 1995.

[26] A. Nayebi, S. Meraji, A. Shamaei, and H. Sarbazi-Azad, "XMulator: A listener-based integrated simulation platform for interconnection networks," In *Asia International Conference on Modeling & Simulation (AMS)*, pp. 128-132, 2007.