Microprocessors and Microsystems 52 (2017) 23–33

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

CAP-W: Congestion-aware platform for wireless-based network-on-chip in many-core era☆

Amin Rezaei[a,*], Masoud Daneshtalab[b], Dan Zhao[c]

[a] Department of Electrical Engineering and Computer Science, Northwestern University (NU), Evanston, USA
[b] Department of Electronic Systems, Royal Institute of Technology (KTH), Stockholm, Sweden
[c] Department of Computer Science, Old Dominion University (ODU), Norfolk, USA

ARTICLE INFO

Article history:
Received 30 September 2016
Revised 5 May 2017
Accepted 24 May 2017
Available online 26 May 2017

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

ABSTRACT

In order to fulfill the ever-increasing demand for high-speed and high-bandwidth, wireless-based MCSoC is presented based on a NoC communication infrastructure. Inspiring the separation between the communication and the computation demands as well as providing the flexible topology configurations, makes wireless-based NoC a promising future MCSoC architecture. However, congestion occurrence in wireless routers reduces the benefit of high-speed wireless links and significantly increases the network latency. Therefore, in this paper, a congestion-aware platform, named CAP-W, is introduced for wireless-based NoC in order to reduce congestion in the network and especially over wireless routers. The triple-layer platform of CAP-W is composed of mapping, migration, and routing layers. In order to minimize the congestion probability, the mapping layer is responsible for selecting the suitable free core as the first candidate, finding the suitable first task to be mapped onto the selected core, and allocating other tasks with respect to contiguity. Considering dynamic variation of application behaviors, the migration layer modifies the primary task mapping to improve congestion situation. Furthermore, the routing layer balances utilization of wired and wireless networks by separating short-distance and long-distance communications. Experimental results show meaningful gain in congestion control of wireless-based NoC compared to state-of-the-art works.

1. Introduction

Nowadays, commercial Many-Core System-on-Chips (MCSoCs) are available based on Network-on-Chip (NoC) [1] communication infrastructure [2]. It is also predicted that upcoming MCSoCs will progressively continue operating on completely new principles and novel NoC-based architectures. In comparison with the traditional or hierarchical bus interconnection networks, mesh-based NoC provides more regular, scalable, and flexible framework. Although mesh-based NoC architecture has many advantages, its multi-hop nature places a negative impact on latency of the system. This will become even more challengeable when the network size will be increased by technology scaling. To address the above problem, alternative technologies such as wireless, 3D, and photonic NoC have been emerged [3–5].

Since wireless NoC provides high-speed as well as high-bandwidth and flexible topology configurations, this emerging technology is gaining momentum to be a promising future on-chip interconnection paradigm. However, wireless transceivers along with associated on-chip antennas impose extensive area and power overheads into the system. Accordingly, a hybrid wireless NoC has been proposed using both wired and wireless links [6,7] rather than a single NoC spanning the entire system. Besides, a Hierarchical Wireless NoC (HWNoC) architecture has been introduced where the system is divided into a two-level network [8]. The wired network is responsible for handling the short-distance communications, while the wireless network is capable of conducting the long-distance communications by almost single-hop wireless links. Also, a Wireless Router (i.e. a router equipped with a wireless interface, WR) placement has been proposed for HWNoC to allocate optimal number of WRs across the network [9].

On the other hand, NoC-based MCSoCs face fully-dynamic workloads where diverse applications, as sets of communicating tasks, enter and leave the system at run-time. The overall performance of a NoC-based MCSoC is in a close correlation with network congestion [10]. Congestion not only increases the network latency severely [11], but also raises the network power con-

☆ This paper is the extension of the paper entitled "Efficient congestion-aware scheme for wireless on-chip networks," presented in the Proceedings of 24th International Conference on Parallel, Distributed and Network-Based Computing (PDP-2016).

* Corresponding author.
E-mail address: me@aminrezaei.com (A. Rezaei).

http://dx.doi.org/10.1016/j.micpro.2017.05.014
0141-9331/

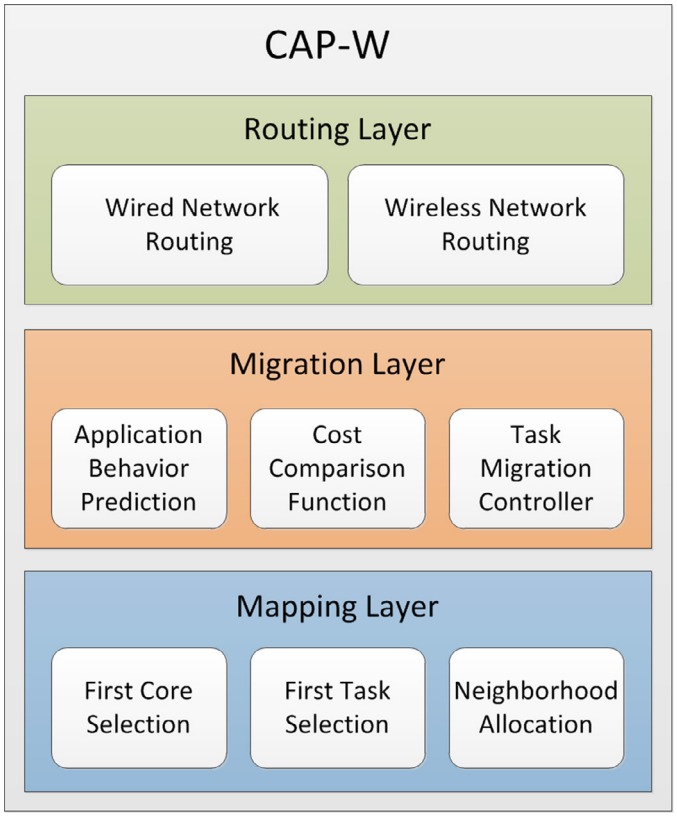

Fig. 1. Triple-layer model of CAP-W.

sumption significantly [12]. Since each WR is shared by a cluster of cores, WRs are more vulnerable to congestion than Conventional Routers (CRs). This becomes to a critical bottleneck especially when the number of WRs that can be integrated into a single chip is limited due to high area and power overheads of the WRs. Moreover, we cannot change the number and the placement of WRs after fabricating the chip.

Along similar lines, in this paper, a Congestion-Aware Platform for Wireless NoC (CAP-W) is introduced in order to reduce congestion in the network and especially over WRs. As shown in Fig. 1, CAP-W consists of three main layers.

Mapping layer is responsible to dynamically map the incoming applications to the available (i.e. free) cores of the system. The mapping layer consists of three main functions. First, it selects the suitable free core as the first candidate to start mapping. Second, it finds the suitable first task to be mapped onto the first core. Third, it allocates other tasks with respect to contiguity.

Migration layer considers the dynamic variation of application behavior. It consists of three main components. First, an application behavior prediction algorithm. Second, a cost comparison function for initiating the task migration. Third, a task migration controller to manage the migration process.

Routing layer separates the wired and wireless network routing in order to balance the usability of WRs.

The rest of the paper is arranged as follows. Section 2 addresses backgrounds and motivations. A dynamic application mapping approach for HWNoC is presented in Section 3. Section 4 proposes a self-aware migration scheme for HWNoC. Furthermore, an adaptive routing algorithm for HWNoC is suggested in Section 5. Lastly, simulation results and conclusion are given in Sections 6 and 7 respectively.

2. 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 [3]. A low Terahertz (324 GHz) frequency generator is realized in 90 nm CMOS [13]. Moreover a signal source operating near 410 GHz that is fabricated using low-leakage transistors in a 45 nm digital CMOS technology is reported [14]. Based on these techniques, the output power level of the on-chip millimeter-wave generator can be as high as −1.4 dBm in the 32 nm CMOS technology, which is large enough for on-chip short distance communication [15]. 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 500 GHz. 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 10 Gb/s with low RF power consumption of only 5pJ/bit is commercially available [16].

Since each WR is shared by many cores, 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 many-core systems, such as Nearest Neighbor (NN) and Best Neighbor (BN) are presented [17,18]. 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) [19], the mapping problem is break down into two steps: the region selection and the task allocation. In the region selection step, the algorithm starts from the closest node to the Central Manager (CM) and includes it in the region. Then, the nodes are iteratively added 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. Moreover, in [20] a Dynamic Application Mapping Algorithm (DAMA) is presented and evaluated for HWNoC that is inspired by CAP-W for 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 [21] a lightweight migration mechanism for bus-based MCSoC 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 [22] 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. In [23] a task migration protocol is presented. Task may be migrated at any moment, not requiring migration checkpoints, and its context is also migrated. Also, in [35] a run-time processor allocation mechanism is introduced by monitoring the allocation/de-allocation of the network nodes and remapping the task based on the currently available processing nodes. Furthermore, an efficient Self-Aware task Migration (SAMi) approach is proposed for NoC-based MCSoCs depending on application prediction [36]. However, the proposed task mi-

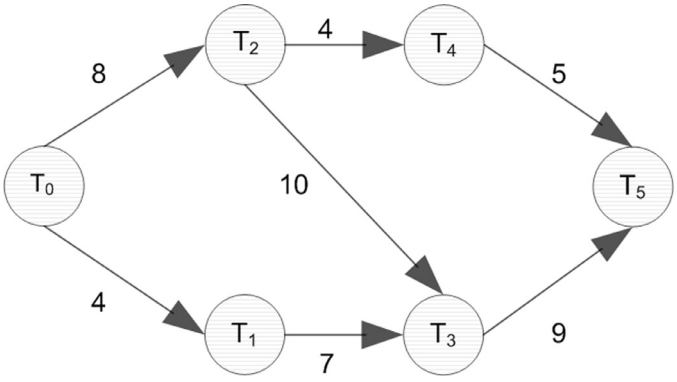

Fig. 2. Task graph of an application with 6 tasks and 7 edges.

gration 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. In [24] a comprehensive reference to routing algorithms as well as discussions of advanced solutions applied to current and next generation of NoC-based MCSoCs is provided. When the shortest paths are congested, sending more packets though them worsen the congestion condition considerably. Therefore, a non-minimal routing algorithm for NoCs that provides a wide range of alternative paths between each pair of source and destination is presented [25]. Moreover, a partitioning method divides the network into logical partitions. All the partitioning methods can be supported by a deterministic routing algorithm. However, in order to increase the performance, the authors in [26] proposed a general minimal and adaptive routing algorithm which is based on the Hamiltonian path and can be applied to all partitioning methods. Again since most of the previous routing algorithms are targeted conventional NoCs, we propose an adaptive routing algorithm based on the separation of wired and wireless networks in order to reduce congestion over WRs.

3. Mapping layer

Two types of congestion can be considered from the 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 a 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 Eq. (1). TG of an application with 6 tasks is shown in Fig. 2. The amount of data transferred from the source task to the destination task is written on the edge.

$$\forall\ t_i \in T,\ \forall\ e_{i,j} \in E,\ A_p = TG(T,\ E) \tag{1}$$

CAP-W's task mapping approach consists of three steps. 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, 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 order to reduce the external congestion. In the following we present each step separately. Without loss of generality, we assume CM is resided to node $n_{0,0}$ in our examples.

3.1. First node selection

The most contiguous area is almost circular [19]. 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 [27].

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 to minimize the number of these nodes while to keep the model almost in the square-shaped. The rectangle models of four running applications are shown in Fig. 3a. For instance, the rectangle of the application 1 has two nodes which do not belong to it ($n_{0,2}$ and $n_{1,2}$). Also the rectangle of the application 3 includes one node ($n_{6,1}$) which is not a part of the application. However, they are the best fitting rectangles in order to stay close to square-shaped.

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. 3b 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. 3b. 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 square-shaped.

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. 3b, two WRs ($n_{4,7}$ 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 one closer 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. The two candidates for the first node of the application 5 are shown in Fig. 3b. CAP-W's 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. Fig. 4 shows flowchart of CAP-W's first node selection.

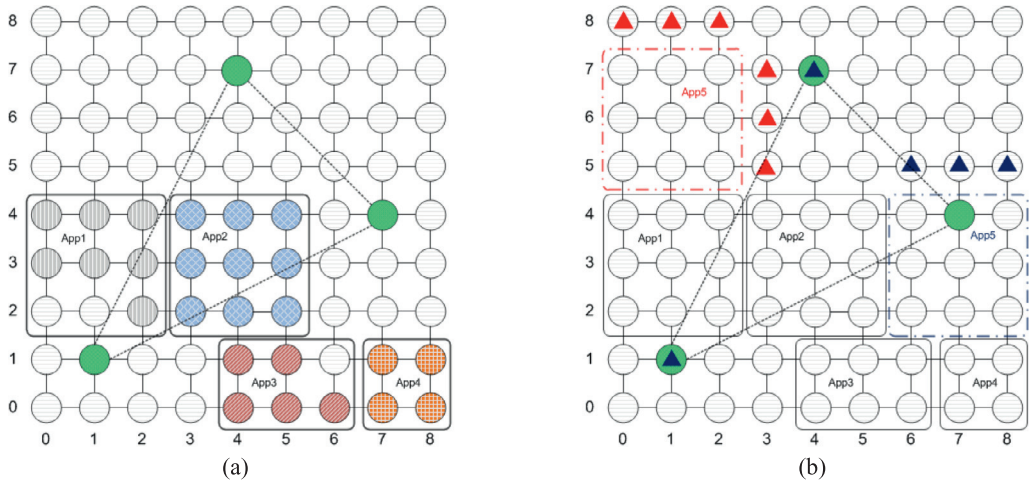

Fig. 3. CAP-W's square factor calculation example.

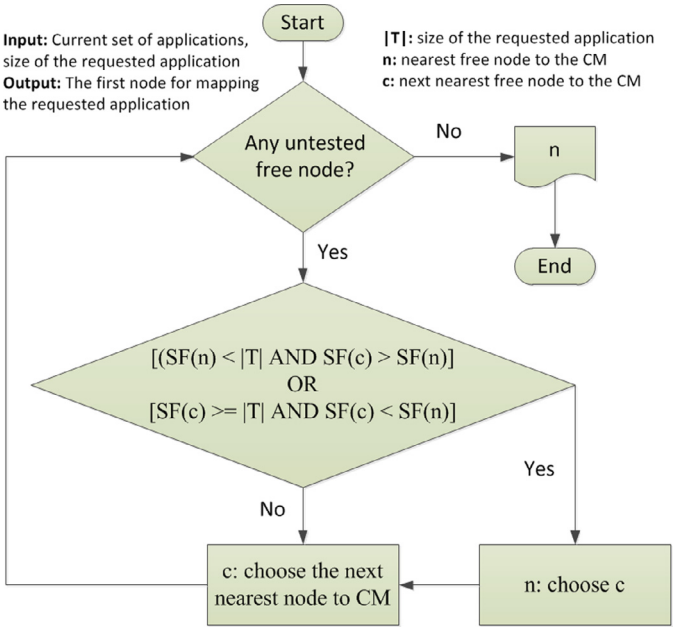

Fig. 4. CAP-W's first node selection flowchart.

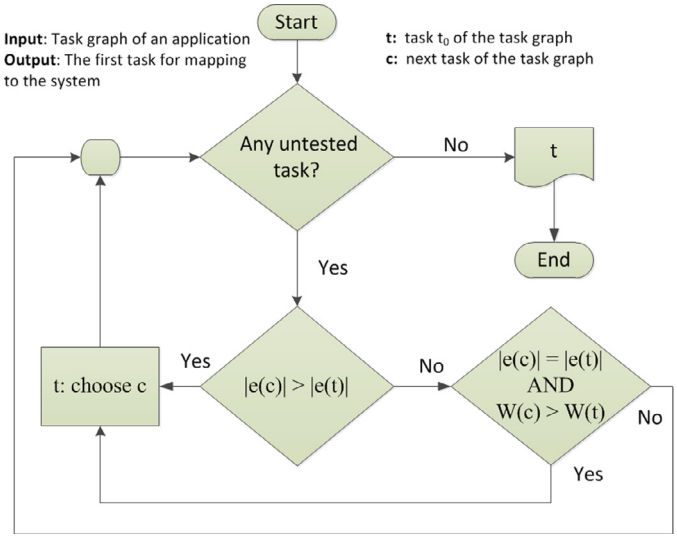

Fig. 5. CAP-W's first task selection flowchart.

3.2. First task selection

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 be controlled by almost one-hop links, which reduces the internal 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. 2, both tasks t_3 and t_2 have 3 edges. Accordingly, since the total communication weight of t_3 is more than that of t_2 (26 vs. 22), t_3 is selected as the first task to be mapped onto the first node. Fig. 5 demonstrates flowchart of CAP-W's first task selection.

3.3. 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.

For example considering the application 5 in Fig. 3b, after mapping the first task to the first node, which is $n_{7,3}$, 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. 6. As a result, CAP-W's task mapping approach maps the communicating tasks onto the closest neighborhood, while keeping the mapped area as close to square as possible. As can be seen, the application not only is mapped onto a contiguous region without any internal congestion, but also imposes no external congestion on its neighboring applications due to the rectangularity of the mapped area. CAP-W's neighborhood allocation flowchart is shown in Fig. 7.

4. Migration layer

Significant variation of application behavior places an upper-bound on improving the performance by primary task mapping. This makes some mapping strategies take behavior variation into

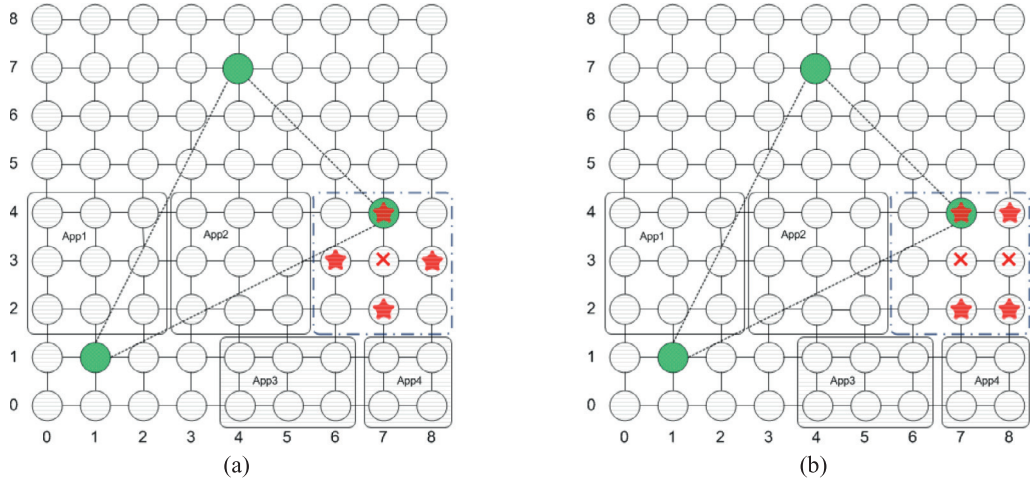

Fig. 6. CAP-W's neighborhood allocation example.

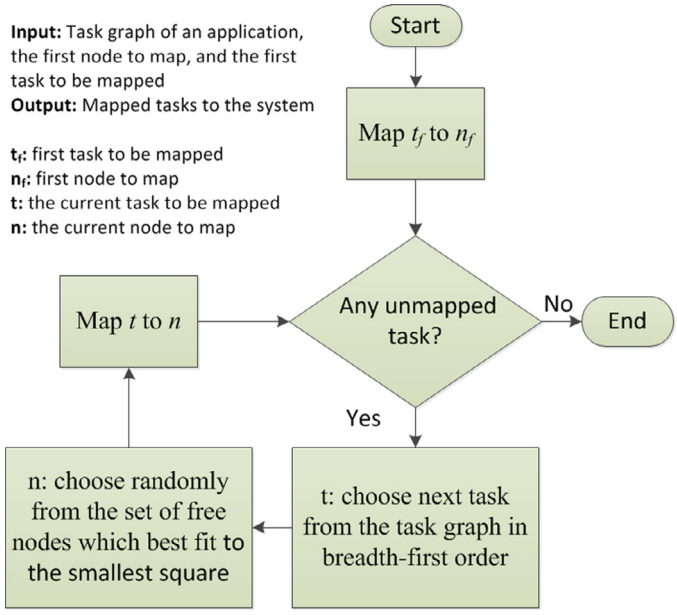

Fig. 7. CAP-W's neighborhood allocation flowchart.

consideration [28]. Task migration is a dynamic task re-mapping mechanism which follows real-time variation of workload behavior. Task migration in a HWNoC is described as transferring a task from the source core where it is currently running to a destination core and then resuming its execution there in such a way that some system performance objectives are improved. The main goal in this paper is improving congestion.

CAP-W's task migration scheme consists of three parts. First, an application behavior prediction algorithm. Second, a cost comparison function for initiating the task migration. Third, a task migration controller to manage the migration process.

HWNoC is represented by an architecture graph $AG(C, L)$. The AG contains a set of cores $c_i \in C$, which are connected together through unidirectional links $l_k \in L$. Each $c_i \in C$ may has a running weighted task $(t_i, w_i) \in c_i$ on it. The weight is defined as the communication demand (i.e. the edge of TG in Section 3) of each task. Task migration is defined as transferring the running weighted task $(t_s, w_s) \in c_s$ from source core $c_s \in C$ to destination core $c_d \in C$ and then resuming its execution. In the following we present each part separately.

4.1. Application behavior prediction

A general approach for predicting the future is to capture the past behaviors. Most existing works rely on some counters for capturing past behaviors because of the simplicity and low area overhead of such counters [29]. There are two predictors supported by CAP-W: short-term and long-term predictors. The short-term predictor anticipates based on the recent information stored in $counter_{i,k}$, where $(t_i, w_i) \in c_i$ and $(t_k, w_k) \in c_k$ are the tasks communicating with each other. On the contrary, the long-term predictor anticipates based on the pattern of communication experiences. The prediction function is defined as follows:

$$P_t(i,k) = \begin{cases} A_{t-1}(i,k), & sel(i,k) = 0 \\ table_{i,k}\left(history_{i,k}\right), & sel(i,k) = 1 \end{cases} \quad (2)$$

Where $A_t(i, k)$ is the actual traffic and $P_t(i, k)$ is the predicted traffic from task $(t_i, w_i) \in c_i$ to task $(t_k, w_k) \in c_k$ at the t-th interval. Also, $history_{i,k}$ is the record of the communication pattern between tasks $(t_i, w_i) \in c_i$ and $(t_k, w_k) \in c_k$ while $table_{i,k}$ is the long-term prediction table, in which each entry is indexed by $history_{i,k}$ and includes a prediction rate. The prediction rate traces the amount of data transmitted when this pattern was encountered last time. If the same pattern appears again, the traced value is used as the prediction rate. The anticipation either comes from the short-term predictor or the long-term predictor, decided by a selector function sel. The selector function is designed according to the system requirements.

4.2. Cost comparison function

We define a trigger called Core Congestion (CC) in order to control congestion of the system. The migration starts when the amount of packets received by a core in the network reaches the core congestion threshold (th_c). The value for th_c is defined adaptively based on the average traffic of the cores and prediction of the application behavior.

A cost function is needed in order to determine the best destination core for task migration. For this purpose, the destination core is the one which has the minimum congestion among all other cores while satisfying the neighborhood allocation constraint after the task migration. (i.e. it fits into the smallest square with the first node.) If there is more than one candidate as the destination, one of them is randomly selected. Note that the first node of each application is not considered for migration. Simply speaking, based on the application behavior and core congestion trigger, the

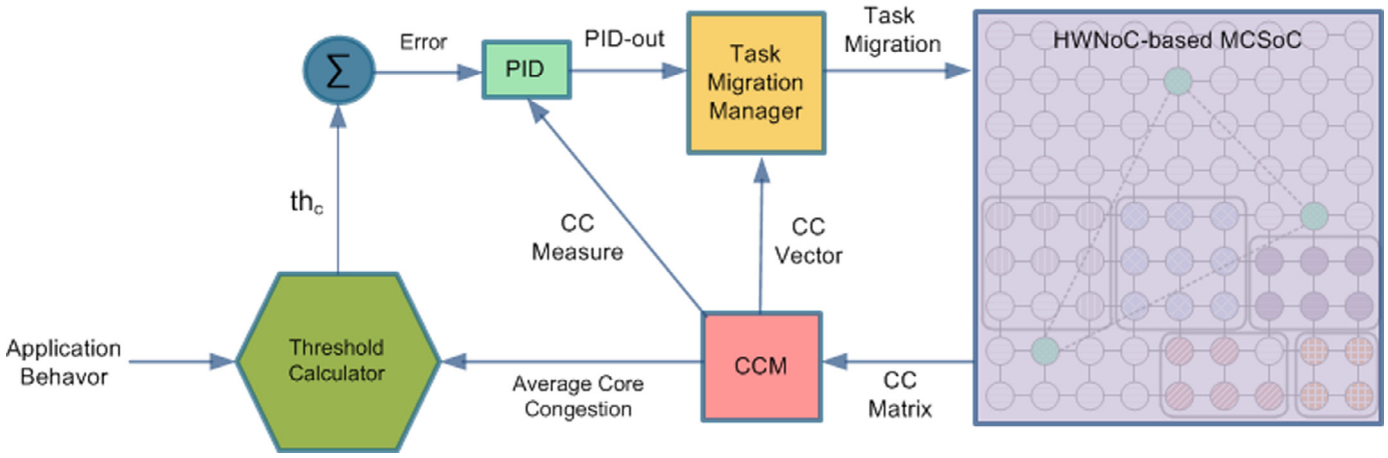

Fig. 8. CAP-W's congestion control platform.

migration cost function modifies the random decision that is made in neighborhood allocation phase.

4.3. Congestion control platform

Basically each controller compares the system output with a target value. After comparison, it manipulates the system actuators to minimize the error. The controller policy to tune the actuators strongly depends on the dynamic model of the target system and the system robustness against error disturbance. The dynamic model defines how the system reacts to the inputs including actuations and other inputs. The system robustness is defined as the system stability against overshooting of the output values from the target intended output. CM is responsible to manage the migration process. The proposed congestion control platform for HWNoC is depicted in Fig. 8.

4.1.1. Congestion meter

Each core measures the traffic dynamically by calculating the moving average of packet flow in every link of its router. Then, the congestion level of each router is transferred to CM. Then, CM sends these information to Core Congestion Meter (CCM). CCM computes average core congestion level and sends it to the threshold calculator.

4.1.2. Threshold calculator

Threshold Calculator (TC) calculates the amount of th_c based on the average core congestion and also prediction of the application behavior.

4.1.3. PID controller

A Proportional Integral Derivative (PID) controller for actuator manipulation is employed. The general formula for the PID controller is as follows:

$$PID_{out}(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt} \quad (3)$$

Where $PID_{out}(t), e(t)$, K_p, K_i, and K_d are the controller output, error, proportional gain, integral gain, and derivative gain, respectively. The proportional part determines how fast or aggressive the controller reacts to changes in the input signal. The main function of the integral part is to ensure that the process output agrees with the set point value in steady state. The derivative part determines how the system reacts to changes in the reference value; It also enhances stability in the system [30].

4.1.4. Task migration manager

Task Migration Manager (TMM) performs the task migration based on the information from controller outputs and congestion vectors. When a core marked as congested by the PID output, the TMM finds the best destination core for the task migration based on the congestion vectors. (i.e. the cost comparison function of the previous part).

In order to lower the overhead of task migration strategy, it is implemented based on MCSoC Message Passing Interface [31], in which task mapping is independent of task re-mapping. By changing task mapping table, task is remapped to another core. Then task state information is transferred. Hence, the migrated task can restore execution on a different core. The task migration contributes less communication overhead because task state information excludes task code. Therefore, after choosing the destination, TMM transfers state information of the chosen task from the source core to the destination core. The task then resumes its execution there. Hence, remapping a task of application might not affect the application execution since the communication handles through the message passing interface regardless of the new location of the migrated task.

5. CAP-W routing layer

In HWNoC, the communication can be 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. For network efficiency, HWNoC is partitioned in a way that any core within a region has the minimum hop-count towards the WR of that region than the WRs of the other regions. For borderline cases that one core may have the same hop-count from two or more WRs, the core will be randomly assigned to one of the candidate regions. Fig. 9 shows the partitioning of 81-core HWNoC into three regions. 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-count and congestion is used in order to select the efficient path.

Fig. 10 represents CAP-W's routing algorithm. 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 WRs.

$$\delta = C \times u \quad (4)$$

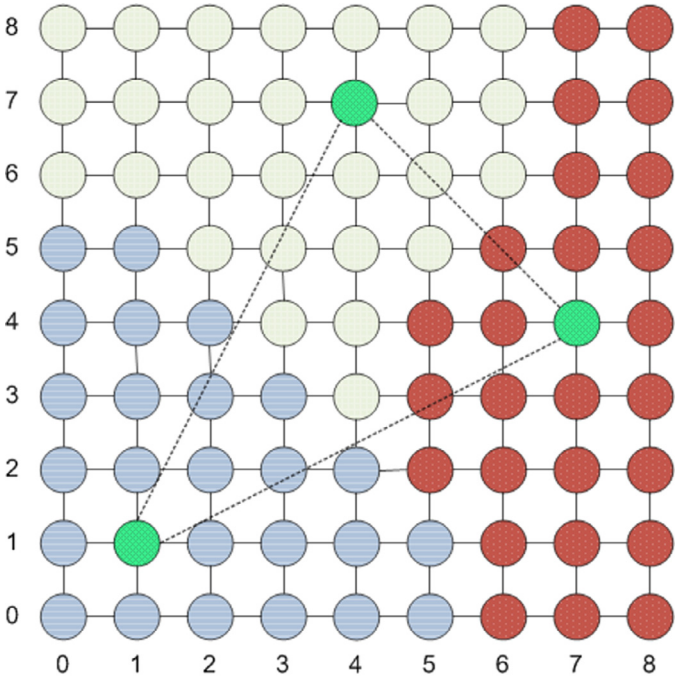

Fig. 9. CAP-W's partitioning example.

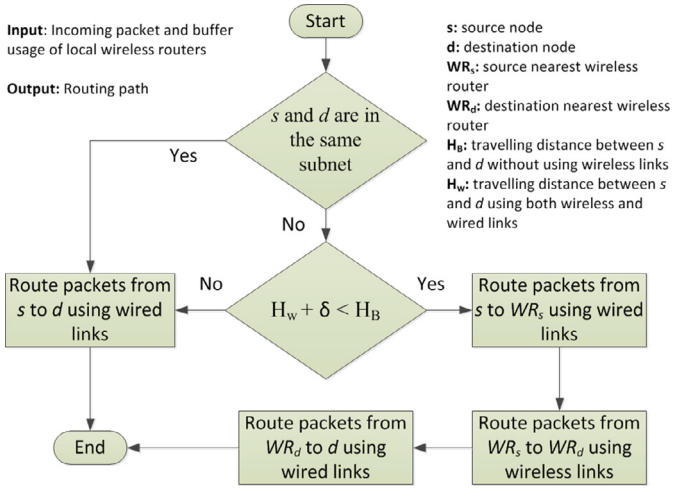

Fig. 10. CAP-W's routing flowchart.

As it is shown in Eq. (4), 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.

One of the principal subjects should be addressed in networks using wormhole switching is the deadlock avoidance. Although using a Dimension Order Routing (DOR) like XY routing, 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. 11a. 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. 11b). One of them is for traffic transmission using nearest WR while the other one is utilized for either the wired network or traffic transmission of the WR to the destination node.

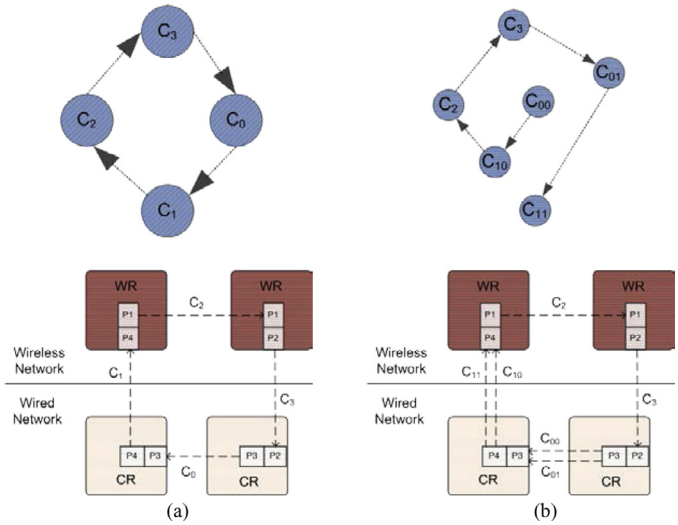

Fig. 11. A deadlock prone situation in CAP-W (a) Without using virtual channels (b) By using virtual channels.

6. Experimental results

In this section, we assess the impact of CAP-W platform on improving the congestion of the system. Several set of applications with 4 to 35 tasks are generated using TGG [32] 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's routing algorithm, some application benchmark suites selected from SPLASH-2 [33] are used. Experiments are performed using XMulator [34] 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 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 CM, 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.

6.1. Evaluation metrics

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.

$$AMD_{map(A)} = \frac{\sum_{\forall\ e_{i,j} \in E} MD\big(map(t_i),\ map(t_j)\big)}{|E|} \tag{5}$$

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,

Table 1
Latency evaluation for different mapping algorithms.

	NN [17]	BN [18]	INC [19]	CAP-W
AWD	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

averaged by the total communication weights.

$$AWMD_{map(A)} = \frac{\sum_{\forall\ e_{i,j} \in E} W_{i,j} \times MD\big(map(t_i),\ map(t_j)\big)}{\sum W_{i,j}} \quad (6)$$

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:

$$MRD_{map(A)} = \frac{\sum_{\forall\ t_i,\ t_j \in T} MD\big(map(t_i),\ map(t_j)\big)}{\binom{|T|}{2}} \quad (7)$$

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:

$$MRD_{SQ(|T|)} = \frac{2\ \times \sqrt{|T|}}{3} \quad (8)$$

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:

$$NMRD_{map(A)} = 1 + \frac{\left|MRD_{map(A)} - MRD_{SQ(|T|)}\right|}{\left|MRD_{SQ(|T|)}\right|} \quad (9)$$

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.

6.2. Latency evaluation

The packet latency and values of different evaluation are summarized in Table 1. 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 BN algorithm by 40% and 20% reduction in external and internal congestion factors, respectively. It also obtains 50% gain over INC algorithm in internal congestion and more than 15% in external congestion. In addition, CAP-W has 25% gain in average for all evaluation aspects in comparison with NN algorithm.

As shown in [19], 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 (MD). The experiments have been run for different algorithms in the injection rate of 2/3 λ_{full}. As depicted in Fig. 12, more than 80% of the packets are delivered by one hop distance using CAP-W scheme.

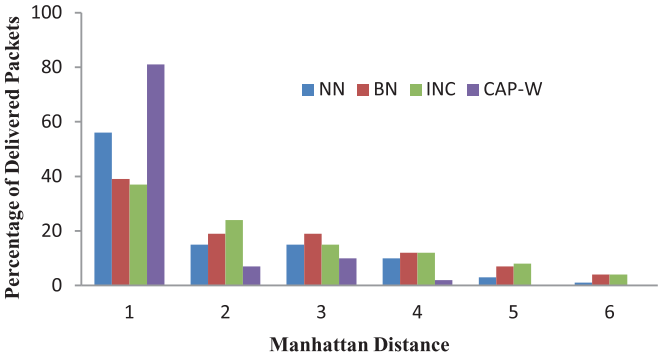

Fig. 12. Percentage of delivered packets in different path lengths.

Table 2
Time complexity of task migration scheme.

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

In addition, Fig. 13 shows the average hop-count and the average packet latency among the baseline NoC without any WR integration, HWNoC using the XY routing and DAMA mapping [20], and HWNoC equipped with CAP-W. As it can be seen, although HWNoC with XY and DAMA outperforms the HWNoC using CAP-W in average hop-count, it suffers from network latency caused by congestion over WRs.

6.3. Time complexity assessment

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. 14a. The injected rate is set to 3/4 λ_{full}. Furthermore, the time complexity of different mapping for applications with 8 tasks is presented in Fig. 14b, 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. 14, all mapping algorithms scale well when the injection rate is increased.

Furthermore, Table 2 shows the average number of clock cycles required for all the task migrations taken place in each injection rate. From Fig. 14b and Table 2, the overhead time of task migration is negligible compared to mapping time complexity, about 0.1× in the worst case scenario.

6.4. System utilization measurement

System utilization is another important factor has been analyzed among the baseline NoC, HWNoC with XY and DAMA [20], and HWNoC with CAP-W. Note that the system utilization is based on the number of tasks that can be mapped on the cores which communicate with each other without dropping due to the high congestion. As shown in Fig. 15, CAP-W increases the average system utilization compared to the baseline NoC and HWNoC 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. 15. CAP-W also increases the maximum utilization compared to the baseline NoC and HWNoC with XY and DAMA. As can be noticed, the proposed platform cannot reach 100% utilization because area fragmentation usually occurs due to

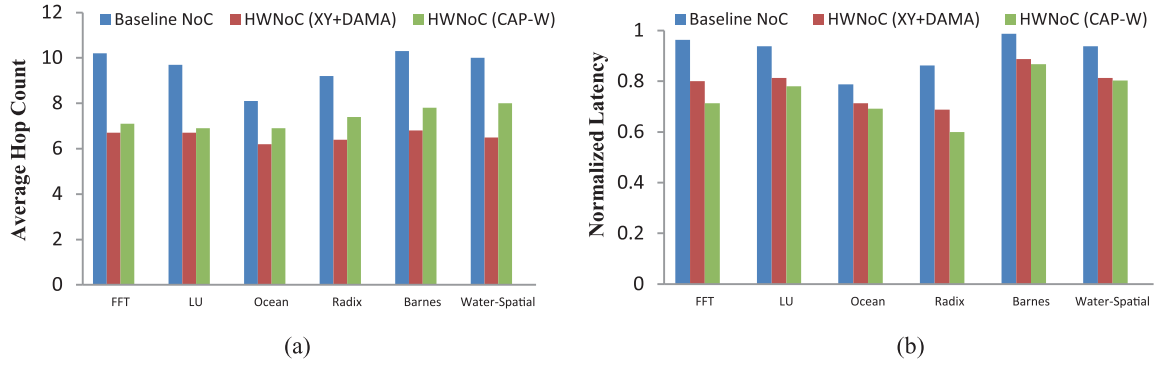

Fig. 13. (a) Hop-count comparison (b) Latency comparison.

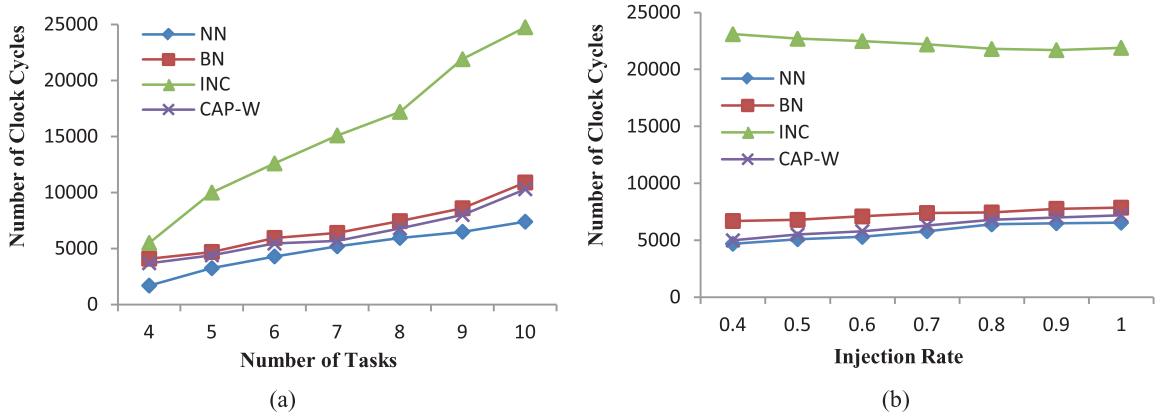

Fig. 14. Time complexity of mapping algorithms over (a) Application sizes (b) Different injection rates.

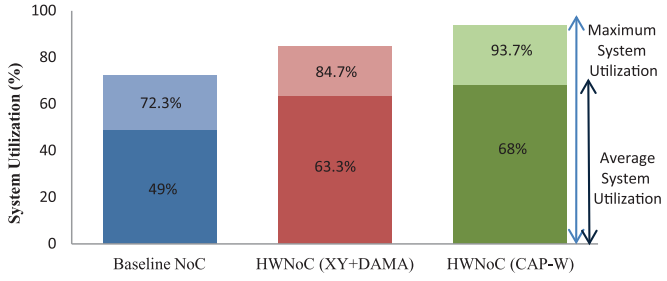

Fig. 15. System utilization comparison.

the dynamic mapping policy. (i.e. when applications does not exactly fit onto the many-core system.)

7. Conclusion

In this paper, we proposed an efficient congestion-aware platform, CAP-W, for wireless-based MCSoC. CAP-W targets at reducing internal and external congestions, and includes three main layers. First, a dynamic task mapping approach that tries to minimize congestion probability; Second a task migration strategy that considers dynamic variation of application behaviors; Third, an adaptive routing algorithm that balances utilization of wired and wireless networks. 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. For future work, task migration overhead can be reduced using hierarchical managing scheme [37].

References

[1] L. Benini, G.D. Micheli, Networks on chips: a new SoC paradigm, IEEE Comput. 35 (1) (2002) 70–78.
[2] “Adapteva, Inc.” [Online]. Available: http://www.adapteva.com/; “Arteris, Inc.” [Online]. Available: http://www.arteris.com/; “Sonics, Inc.” [Online]. Available: http://sonicsinc.com/.
[3] M.F. Chang, J. Cong, A. Kaplan, M. Naik, G. Reinman, E. Socher, 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), 2008, pp. 191–202.
[4] V.F. Pavlidis, E.G. Friedman, 3-D topologies for networks-on-chip, IEEE Trans. Very Large Scale Integr. 15 (10) (2007) 1081–1090.
[5] A. Shacham, K. Bergman, S. Member, L.P. Carloni, Photonic networks-on-chip for future generations of chip multiprocessors, IEEE Trans. Comput. 57 (9) (2008) 1246–1260.
[6] A. Ganguly, K. Chang, S. Deb, P.P. Pande, B. Belzer, C. Teuscher, Scalable hybrid wireless network-on-chip architectures for multicore systems, IEEE Trans. Comput. 60 (10) (2011) 1485–1502.
[7] S. Dep, K. Chang, X. Yu, S.P. Sah, M. Cosic, A. Ganguly, P.P. Pande, B. Belzer, D. Heo, Design of an energy-efficient CMOS-compatible NoC architecture with millimeter-wave wireless interconnects, IEEE Trans. Comput. 62 (12) (2013) 2382–2396.
[8] A. Rezaei, F. Safaei, M. Daneshtalab, H. Tenhunen, HiWA: a hierarchical wireless network-on-chip architecture, in: Proceedings of IEEE International High Performance Computing & Simulation (HPCS), 2014, pp. 499–505.
[9] A. Rezaei, M. Daneshtalab, F. Safaei, D. Zhao, Hierarchical approach for hybrid wireless network-on-chip in many-core era, Els. Int. J. Comput. Electr. Eng. 51 (C) (2016) 225–234.
[10] C.L. Chou, R. Marculescu, Contention-aware application mapping for network-on-chip communication architectures, in: IEEE International Conference on Computer Design (ICCD), 2008, pp. 164–169.
[11] J.W. Brand, C. Ciordas, K. Goossens, T. Basten, Congestion-controlled best-effort communication for networks-on-chip, in: Proceedings of Design, Automation and Test in Europe (DATE), 2007, pp. 1–6.
[12] S. Ma, N.E. Jerger, 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), 2011, pp. 413–424.
[13] D. Huang, T. LaRocca, M.C. Chang, L. Samoska, A. Fung, R. Campbell, M. Andrews, Terahertz CMOS frequency generator using linear superposition technique, IEEE J. Solid State Circ. 43 (12) (2008) 2730–2738.

[14] E. Seok, C. Cao, D. Shim, D.J. Arenas, D.B. Tanner, C. Hung, K.K.O, A 410 GHz CMOS push-push oscillator with an on-chip patch antenna, in: IEEE International Solid-State Circuits Conference (ISSCC), 2008, pp. 472–629.
[15] S.B. Lee, S.W. Tam, I. Pefkianakis, S. Lu, M.F. Chang, C. Guo, G. Reinman, C. Peng, M. Naik, L. Zhang, J. Cong, A scalable micro wireless interconnect structure for CMPs, in: Proceedings of the International Conference on Mobile Computing and Networking (MobiCom), 2009, pp. 217–228.
[16] W. Green, M. Rooks, L. Sekaric, Y. Vlasov, Ultra-compact, low RF power, 10 Gb/s silicon Mach–Zehnder modulator, Optics Exp. 15 (25) (2007) 17106–17113.
[17] E.L. Carvalho, N.L.V. Calazans, F.G. Moraes, Heuristics for dynamic task mapping in NoC-based heterogeneous MPSoCs, in: IEEE/IFIP International Workshop on Rapid System Prototyping (RSP), 2007, pp. 34–40.
[18] E.L. Carvalho, N.L.V. Calazans, F.G. Moraes, Dynamic task mapping for MPSoCs, IEEE Des. Test Comput. 27 (5) (2010) 26–35.
[19] C.L. Chou, U.Y. Ogras, 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, 27, 2008, pp. 1866–1879.
[20] A. Rezaei, M. Daneshtalab, D. Zhao, F. Safaei, X. Wang, M. Ebrahimi, Dynamic application mapping algorithm for wireless network-on-chip, in: Proceedings of IEEE Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP), 2015, pp. 421–424.
[21] S. Bertozzi, A. Acquaviva, D. Bertozzi, A. Poggiali, Supporting task migration in multi-processor systems-on-chip: a feasibility study, in: Proceedings of Design, Automation and Test in Europe (DATE), 2006, pp. 1–6.
[22] B. Goodarzi, 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), 2011, pp. 463–469.
[23] F.G. Moraes, G.A. Madalozzo, G.M. Castilhos, E.A. Carara, Proposal and evaluation of a task migration protocol for NoC-based MPSoCs, IEEE Int. Symp. Circ. Syst. (ISCAS) (2012) 644–647.
[24] M. Palesi, M. Daneshtalab (Eds.), Routing Algorithms in Networks-on-Chip, Springer, 2014.
[25] M. Ebrahimi, M. Daneshtalab, F. Farahnakian, J. Plosila, P. Liljeberg, M. Palesi, H. Tenhunen, HARAQ: congestion-aware learning model for highly adaptive routing algorithm in on-chip networks, in: Proceedings of International Symposium on Networks-on-Chip (NoCS), 2012, pp. 19–26.
[26] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, J. Plosila, J. Flich, H. Tenhunen, Path-based partitioning methods for 3D networks-on-chip with minimal adaptive routing, IEEE Trans. Comput. 63 (3) (2014) 718–733.
[27] A. Rezaei, M. Daneshtalab, M. Palesi, D. Zhao, Efficient congestion-aware scheme for wireless On-Chip networks, in: Proceedings of IEEE Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP), 2016, pp. 742–749.
[28] C. Wang, L. Yu, L. Liu, T Chen, Packet triggered prediction based task migration for network-on-chip, in: Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), 2012, pp. 491–498.
[29] Y. Huang, K.-K. Chou, C.-T. King, S.-Y. Tseng, NTPT: on the end-to-end traffic prediction in the on-chip networks, in: Design Automation Conference (DAC), 2010, pp. 449–452.
[30] S. Holmbacka, S. Lafond, J. Lilius, A PID-controlled power manager for energy efficient web clusters, in: Proceedings of IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC), 2011, pp. 721–728.
[31] F. Fu, S. Sun, X. Hu, J. Song, J. Wang, M. Yu, MMPI: a flexible and efficient multiprocessor message passing interface for NoC-based MPSoC, in: Proceedings of IEEE International SoC Conference (SOCC), 2010, pp. 359–362.
[32] "Task graph generator (TGG)." [Online]. Available: http://sourceforge.net/projects/taskgraphgen/.
[33] S. Woo, The SPLASH-2 programs: characterization and methodological considerations, in: International Symposium on Computer Architecture (ISCA), 1995, pp. 24–36.
[34] A. Nayebi, S. Meraji, A. Shamaei, H. Sarbazi-Azad, XMulator: a listener-based integrated simulation platform for interconnection networks, in: Asia International Conference on Modeling & Simulation (AMS), 2007, pp. 128–132.
[35] M. Modarressi, M. Asadinia, H. Sarbazi-Azad, Using task migration to improve non-contiguous processor allocation in NoC-based CMPs, Els. J. Syst. Arch. 59 (7) (2013) 468–481.
[36] A. Rezaei, M. Daneshtalab, D. Zhao, M. Modarressi, SAMi: Self-aware migration approach for congestion reduction in NoC-based MCSoC, in: Proceedings of IEEE International System-on-Chip Conference (SOCC), 2016, pp. 145–150.
[37] A. Rezaei, D. Zhao, M. Daneshtalab, H. Zhou, Multi-objective task mapping approach for wireless NoC in dark silicon age, in: Proceedings of IEEE Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), 2017, pp. 589–592.

Amin Rezaei is currently pursuing his Ph.D. studies at Northwestern University, the USA. He received his B.Sc. degree in Computer Engineering from University of Isfahan, Iran, in 2011 and two M.Sc. degrees one in Computer Engineering from Shahid Beheshti University, Iran, in 2014 and the other in Computer Science from University of Louisiana at Lafayette, the USA, in 2016. His main research interests include Parallel and Heterogeneous Computing, and Multi & Many Core System-on-Chips.

Masoud Daneshtalab is currently Senior Lecturer (Associate Professor) at Mälardalen University and Researcher at KTH Royal Institute of Technology, Sweden. Before that he was Lecturer and Project Manager at University of Turku, Finland. He has been appointed as Associate Editors of Elsevier Journals of Computers & Electrical Engineering and Microprocessors and Microsystems, and in the Editorial Board of The Scientific World Journal, IJDST, IJARAS, IJERTCS, and IJDATICS. He is also in the Euromicro's board of directors since 2016 while representing Sweden in the management committee of the ICT COST Actions IC1202 (TACLe). His research interests include Interconnection Networks, Many-Core and Reconfigurable Systems, and Neuromorphic Architecture. He has published 1 book, 4 book chapters, and over 180 refereed international journals and conference papers.

Dan Zhao is currently an Associate Professor Computer Science at Old Dominion University, the USA. She gained her M.Sc. and Ph.D. from the CSE Department, State University of New York at Buffalo, the USA, in 2001 and 2004, respectively. She received the NSF Career Development Award in 2009 and JSPS Fellowship Award in 2006. Previously she was an Associate Professor in the Center for Advanced Computer Studies at the University of Louisiana at Lafayette, the USA.