Multi-Objective Routing Algorithm for dynamic communications mapping in NoC-based heterogeneous MPSoCs

M. K. Benhaoua[1, 2], A. E. H. Benyamina[1]

1. Department of Computer Science, University of Oran – Es Senia, BP 1524, EL M'Naouer, Oran, Algeria
2. University Lille 1, LIFL, CNRS, UMR 8022, F-59650 Villeneuve d'Ascq, France
Email: [1, 2]{mohammed-Kamel.Benhaoua@lifl.fr}, [1]{benyamina.abouelhassen@univ-oran.dz}

Keywords: Multi-Processor Systems-on-Chip (MPSoCs), Network-on-Chip (NoC), Heterogeneous architectures, Dynamic mapping heuristics, Routing algorithm.

1 Introduction

Intensive embedded systems use Multi-Processor Systems-on-Chip (MPSoCs), which provide increased parallelism towards achieving high performance [1], to cope with the limits of a single general purpose processor, increasing computational demands and performance requirements. An MPSoC [2] contains multiple processing elements (PEs) in the same chip. The Network-On-Chip (NoC) has been introduced as a power efficient and scalable interconnection to support communication amongst the PEs [3].

The designer has to map the tasks of the application onto the different processing resources of the MPSoC. Static mapping techniques defines task placement at design-time, having a global view of the MPSoC resources. Such mapping techniques may use complex algorithms to better explore the MPSoC resources towards achieving optimized solutions. However, static mapping is not able to handle the dynamic workload of tasks or applications that need to be loaded into the system at run-time. Dynamic (run-time) mapping techniques are required to handle these varying (dynamic) workloads. Such techniques find placement of tasks on the MPSoC resources at run-time. The latest dynamic mapping approaches try to place the communicating tasks on the nearest available PEs, i.e. close to each other in order to reduce the communication overhead [4, 5]. However, these approaches do not perform well when applications contain a large number of tasks. Further, most of the mapping works reported in the literature uses a deterministic routing algorithm such as XY routing method [6, 7]. However, for a system that needs to handle dynamic workflow, using a dynamic routing method can lead to better results.

Contributions: We present a dynamic Multi-Objective Routing Algorithm (MORA) that reduces the communication costs when compared to often employed routing approaches. The model used for the representation of applications is the master-slave model. This type of model is used to represent the applications that have parallel communicating tasks. The considered heterogeneous MPSoC platform contains two types of PEs: Instruction Set Processors (ISPs) and Reconfigurable Areas (RAs), which execute software and hardware tasks, respectively. Existing techniques use deterministic routing approaches to facilitate the communication. However, most of them do not focus on the adaptive routing (dynamic communications mapping). In our proposed dynamic MORA tries to find the path of communications that has the lowest load (widest bandwidth), resulting in optimized execution time and energy consumption. The dynamic mapping approaches employing MORA routing method lead to performance improvements when compared to approaches employing other routing methods such as XY.

Proposed Multi-Objective Routing Algorithm (MORA)

The reference heuristics including most of the existing dynamic task mapping approaches (e.g., [4, 5, 8, 9]) use deterministic XY routing algorithm to facilitate communication amongst the communicating tasks once they are mapped onto the PEs. Example of such a routing is shown in Figure 1 (a). The figure shows an example NoC where two communicating tasks are mapped on the source and destination nodes (PEs) and they need to communicate with each other. The values mentioned adjacent to the links represent the volumes present in the links, i.e. the number of packets to be transmitted through the links. Figure 1 (a) indicates that in order to transfer a token from the source PE to the destination PE, the packet is first transferred 2 hop distances in X direction and then 2 hop distances in Y direction while following the XY routing mechanism. The packets are sent one by one in the same direction created by the first packet. In Figure 1 (a), the first chosen link in X direction has volume of (150) that is more than the volume (110) present in Y direction. Similarly, the second chosen link in the X direction has more volume than that of the link in Y direction (250 vs. 80). This mechanism routes the packets through a path that incurs high communication costs due to high volumes present in the links chosen for communication, resulting in high communication costs. Thus, choosing such communication paths may incur high communication time and energy consumption.

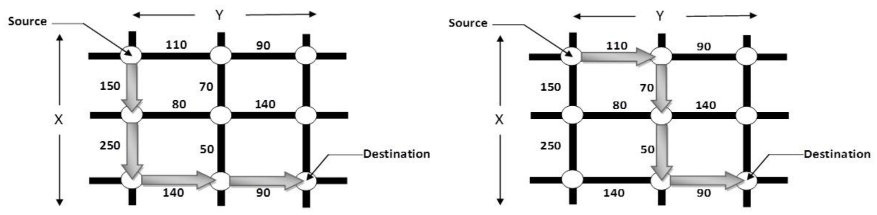
In order to provide efficient communication between the source and destination nodes, an efficient routing strategy needs to be developed. The routing strategy should be able to choose the links with lower volumes at run-time. Figure 1 (b) describes an example for the operation of the proposed dynamic routing algorithm presented in Algorithm 1. Unlike the XY routing, MORA chooses an efficient routing path where the packets are transferred by the links having the lowest loads. The direction to be taken from source to destination PE follows different paths depending upon the location of the PEs and loads in the paths. If x-coordinate of the source (X_{source}) is less than the x-coordinate of the destination (X_{dest}), then the trajectory (path) will be *up to down*; otherwise *down to up*. For down to up, if y-coordinate of the source (Y_{source}) is less than the y-coordinate of the destination (Y_{dest}), then the path will be *left to right*, else *right to left*. For all the different paths, the algorithm chooses the link direction that has the lowest load. For example, Algorithm 2 shows how the lowest loaded link is found in the case of Up_to_Down-Left_to_Right. Depending upon the load values present in the links, the algorithm chooses *left to right* ($X'=X_{source}, Y'=Y_{source}+1$) or *up to down* link, which has lower loads. Similar approach as that of Algorithm 2 is followed for other cases when Up, Down, Left and Right are contained in the calling function. In the case when Y_{source} and Y_{dest} are on the same (i.e. in the same colum) then the direction is Up to Down or Down to Up and there is no evaluation to get the load values on the link. The direction is automatically taken in one of the two directions. Similarly, if X_{source} and X_{dest} are the same (i.e. in the same row) then the link chosen and the direction is Left to right or Right to Left. This kind of links selection towards the destination PE facilitates to choose the lowest loaded links. Once a chosen link becomes more loaded, another less loaded link is chosen for the packet transmission if the source and destination PE are not in the same row or column. Otherwise, the same link gets used. For all the communicating tasks, the packets to be transferred use the same strategy.

Algorithm 1: Multi-Objective Routing Algorithm

```
Input: $X_{source}$, $Y_{source}$, $X_{dest}$, $Y_{dest}$
Output: X', Y'
1: if $X_{source} < X_{dest}$ then //Up to Down
2:     if $Y_{source} < Y_{dest}$ then
3:         Up_to_Down-Left_to_Right($X_{source}$, $Y_{source}$)
4:     else
5:         Up_to_Down-Right_to_Left($X_{source}$, $Y_{source}$)
6:     end if
7: else // Down to Up
8:     if $Y_{source} < Y_{dest}$ then
9:         Down_to_Up-Left_to_Right($X_{source}$, $Y_{source}$)
10:    else
11:        Down_to_Up-Right_to_Left($X_{source}$, $Y_{source}$)
12:    endif
13: end if
14: if $X_{source}=X_{dest}$ then //in the same row
15:    Right_to_Left-Left_to_Right($X_{source}$, $Y_{source}$)
16: end if
17: if $Y_{source}=Y_{dest}$ then // in the same column
18:    Up_to_Down-Down_to_Up($X_{source}$, $Y_{source}$)
19: end if
```

Algorithm 2: Up_to_Down-Left_to_Right

```
Input: $X_{source}$, $Y_{source}$
Output: X', Y'
1: if get_value_Link($X_{source}$,$Y_{source}+1$) < get_value_Link($X_{source}+1$,$Y_{source}$) then
2:     X' $\leftarrow$ $X_{source}$
3:     Y' $\leftarrow$ $Y_{source}+1$ //Left to Right
4: else
5:     X' $\leftarrow$ $X_{source}+1$ //Up to Down
6:     Y' $\leftarrow$ $Y_{source}$
7: end if
```



(a) XY Routing (b) MORA Routing

Figure 1. XY and MORA

Experimental results

• Scenario: four application sets: 1st set - each application having 5 tasks, 2nd set - each application having 10 tasks, 3rd set - each application having 15 tasks, and 4th set - each application having 20 tasks.

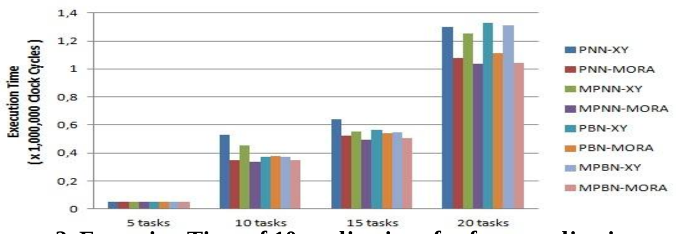


Figure 2. Execution Time of 10 applications for four application sets (Scenario), where each application contains 5, 10, 15 and 20 tasks

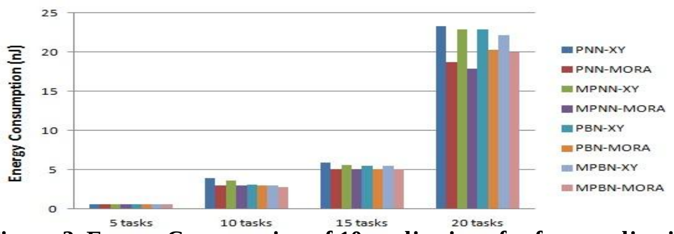


Figure 3. Energy Consumption of 10 applications for four application sets (Scenario), where each application contains 5, 10, 15 and 20 tasks

We have evaluated the performance for large size applications considered in Scenario. Figure 2 shows the execution time gets reduced when MORA routing is employed over the XY routing for all the heuristics. Therefore, our approach provides more savings in total execution time for large size applications. Figure 3 shows energy consumption for four application sets considered in Scenario. It can be observed that the reduction in energy consumption by our approach over the existing approach increases as the number of tasks in the considered applications is increased. Thus, our approach provides better savings for large size applications.

In this paper, we propose heuristic for dynamic communications mapping that considers the placement of communications in order to optimize the overall performance. The mapping technique uses a newly proposed Multi-Objective Routing Algorithm (MORA) to place communications between the tasks. The placement we propose of the communications leads to a better optimization of several performance metrics (time and energy consumption). Experimental results show that the proposed mapping approach provides significant performance improvements when compared to those using XY routing.

References

[1] A. A. Jerraya, H. Tenhunen, W. Wolf, "Guest Editors' Introduction: Multiprocessor Systems-on-Chips", IEEE Computer, Vol. 38, No. 7, 2005, pp. 36–40.

[2] Chip multiprocessor watch, 2008, http://view.eecs.berkeley.Edu/ wiki/Chip Multi Processor Watch.

[3] L. Benini, G. De Micheli, "Networks on chips: a new SoC paradigm", IEEE Computer, Vol. 35, No. 1, 2002, pp. 70–78.

[4] E. Carvalho, N. Calazans, F. Moraes, "Dynamic task mapping for MPSoCs", IEEE Design Test of Computers, Vol. 27, No. 5, 2010, pp. 26–35.

[5] A.K. Singh, T. Srikanthan, A. Kumar, W. Jigang, "Communication-aware heuristics for runtime task mapping on NoC-based MPSoC platforms", Journal of Systems Architecture, Vol. 56, No. 7, 2010, pp. 242 – 255.

[6] A. Mehran, A. Khademzadeh, S. Saeidi, "Dsm: A heuristic dynamic spiral mapping algorithm for network on chip", IEICE Electronics Express, Vol. 5, No. 13, 2008, pp. 464–471.

[7] M. Mandelli, A. Amory, L. Ost, F.G. Moraes, "Multi-task dynamic mapping onto NoC based MPSoCs", In 24th International Symposium on Integrated circuits and systems design (SBCCI'11), 2011, pp. 191–196.

[8] E. Carvalho, F. Moraes, "Congestion-aware task mapping in heterogeneous MPSoCs", In Proc. International symposium on System on Chip (SoC'08), 2008, pp. 1–4.

[9] L. Ost, M. Mandelli, G.M. Almeida, L. Moller, L.S. Indrusiak, G. Sassatelli, P. Benoit, M. Glesner, M. Robert, F. Moraes, "Power-aware dynamic mapping heuristics for NoC-based MPSoCs using a unified model-based approach", ACM Transactions on Embedded Computing Systems, Vol. 12, No. 3, 2013, pp. 1–22.