Adaptive routing framework for network on chip architectures.

Naveed Ul Mustafa, Özcan Özturk, Smail Niar

Adaptive Routing Framework for Network on Chip Architectures

Naveed Ul Mustafa
Department of Computer Engineering, Bilkent University
Ankara, Turkey
naveed.mustafa@bilkent.edu.tr

Ozcan Ozturk
Department of Computer Engineering, Bilkent University
Ankara, Turkey
ozturk@cs.bilkent.edu.tr

Smail Niar
LAMIH-University of Valenciennes
Valenciennes, Cedex 9, France
smniar@univ-valenciennes.fr

ABSTRACT

In this paper we suggest and demonstrate the idea of applying multiple routing algorithms during the execution of a real application mapped on a Network-on-Chip (NoC). Traffic pattern of a real application may change during its execution. As performance of an algorithm depends on the traffic pattern, using the same routing algorithm for the entire span of execution may be inefficient. We study the feasibility of this idea for applications such as SPARSE and MPEG-4 decoder, by applying different routing algorithms. By applying more than one routing algorithms, throughput improves up to 17.37% and 6.74% in the case of SPARSE and MPEG-4 decoder applications, respectively, as compared to the application of single routing algorithm.

CCS Concepts

•Computer systems organization → Interconnection architectures;

Keywords

network-on-chip, routing algorithms, throughput

1. INTRODUCTION

A System on Chip (SoC) consists of multiple Processing Elements (PEs) on a single chip. A NoC is a communication medium among PEs in a SoC [9]. To communicate between a source and a destination node of a NoC, a routing algorithm is needed. Many algorithms have been proposed to establish a routing path with focus on criterion like path setup latency, load balancing, throughput and quality of service. However, a single algorithm cannot give outstanding performance for all traffic patterns.

The traffic pattern of an application is determined by behavior of its active flows. The traffic pattern may change during execution of an application , as tasks mapped on different PEs may not communicate always at the same data rate. For example, a data-producer task may wait for an external event or a computation to be completed before it can send data to a consumer task.

Since, a given routing algorithm may perform well for one traffic pattern but give poor results for others, it is promising to switch routing algorithms with the changing traffic pattern. Performance metrics, like throughput, can be improved by applying multiple routing algorithms on a single application for different execution intervals. In this work, we explore this idea by switching routing algorithms during execution of two real applications, namely SPARSE and MPEG-4 decoder.

The remainder of this paper is organized as follows. In section 2, related work is discussed. Static and dynamic switching of routing algorithms during execution intervals of an application is discussed in section 3 and 4, respectively. We provide experimental evaluation of our idea for SPARSE and MPEG-4 decoder applications in section 5. Conclusion and future work are given in section 6.

2. RELATED WORK

Many of Dynamically Reconfigurable NoCs (DRNoCs) can be reconfigured in terms of architecture or routing algorithm, for example, DyNoc [2], CoNoChi [11] and DRAFT [3]. In order to take care of the dynamically changing architecture in DyNoC, XY routing algorithm is adapted to deal with obstacles created by the dynamic placement and removal of modules. CoNoChi generates a routing table whenever topology changes and tables are then distributed via the network. In DRAFT, a hierarchal level dependent mask is used by each router to determine if the received flit is to be moved upward [7].

All of the above mentioned DRNOCs reconfigure a particular routing algorithm to make it compatible with the modified network architecture. In this work, network architecture is fixed. We used a static mesh network and focused on switching the routing algorithm to take advantage of variations in traffic pattern during execution for improvement in throughput.

In [8], DyAD routing technique is described which selects between two routing algorithms (a deterministic and an adaptive one) based on the congestion threshold and congestion flags. However, we investigate the possibility of making selections among a deterministic algorithm, such as XY routing, and more than one adaptive algorithm (for exam-

RAPIDO '16, January 18 2016, Prague, Czech Republic

DOI: http://dx.doi.org/10.1145/2852339.2852344

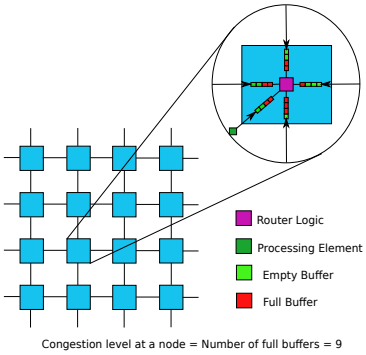

Figure 1: An NxN mesh, congestion level at a node is a measure of occupancy level of input FIFOs.

ple, west first, negative first and north last routing) based on the measurement of congestion level.

3. STATIC SELECTION

Static selection of routing algorithms is performed in two phases. In the first phase, all candidate algorithms are applied on the application individually and values for performance metrics are obtained. In our case, performance metric is throughput and candidate algorithms are XY, west first and negative first.

In the second phase, performance graphs generated by applying candidate algorithms individually on the application are analyzed to find "Switching Points" and "Next Algorithm" for each switching point. "Switching Points" are the time instants at which a new algorithm (other than one currently active) is to be selected. Switching points and next algorithm should be selected in such a way that new performance is higher than that given by candidate algorithms individually.

A major feature of static selection is that switching points and their corresponding next algorithm is determined offline. Furthermore, selection is made at NoC level, which means that all nodes execute same routing algorithm. Static selection is elaborated in section 5 with simulation results for SPARSE and MPEG-4 decoder applications.

4. DYNAMIC SELECTION

Performance of an algorithm depends upon congestion level of the network at a given time and adaptivity of that routing algorithm for a particular application. Therefore it is more logical to make dynamic selection of routing algorithms based on their adaptivity and congestion level of the network. Adaptivity of an algorithm is the number of shortest paths an algorithm allows from a source node to a destination node [5].

Dynamic selection can be made either at the level of regions or nodes. In the case of regional selection, congestion can be measured by using one of the three techniques described in [6]. However, for the sake of simplicity, in this work we select algorithms at the node level. As shown in Figure 1, Congestion Level (CL) at a node is measured as the total length of occupied FIFOs for that node.

For a given application, we first define an Adaptivity List (AL) of algorithms for that application and a set of Congestion Threshold Points (CTPs). AL is a listing of candidate algorithms in such a way that the adaptivity of algorithm i for the given application is less than that for the algorithm *i+1*. N number of CTPs are defined using trial and error method, where N is the total number of candidate algorithms. CTPs are ordered in such a way that $CTP_i < CTP_{i+1}$.

Algorithm 1 describes the selection of routing algorithm for NoC based on CL, CTPs and AL. Application of algorithm 1 is demonstrated in section 5.

Algorithm 1 to switch routing scheme based on congestion level

```
1: procedure ADAPTIVE(CL,CTP,AL,N)
2:     i ← 0
3:     for every simulation cycle do
4:         if i < N then
5:             if CL ≤ CTP[i] then
6:                 Switch routing algorithm to AL[i].
7:             else
8:                 if CTP[i] < CL ≤ CTP[i + 1] then
9:                 Switch routing algorithm to AL[i+1]
10:                    i ← (i+1)
11:                end if
12:            end if
13:        end if
14:    end for
15: end procedure
```

In dynamic selection, different nodes may run different routing algorithms simultaneously. A node can switch to next routing algorithm based on the congestion level information.

5. EXPERIMENTAL EVALUATION

In this work, we used Noxim which is a simulator for NoC, developed using SystemC [4]. In Noxim, we can apply a given routing algorithm on a user defined traffic pattern which is specified as an input traffic trace file. An input traffic trace file must follow the format which is predefined by designers of Noxim. When a given algorithm is applied on an input traffic trace file, Noxim generates values for performance metrics and writes those values in an output trace file which can then be used to generate performance graphs.

We demonstrate the idea of switching routing algorithms during execution of an application by using traffic trace files of SPARSE and MPEG-4 decoder applications. Traffic trace file for SPARSE is extracted from MCSL benchmark suite [10], while traffic trace file for MPEG-4 decoder is based on [1]. These files are processed using MATLAB to make them compatible with the required input format of Noxim.

5.1 Results for SPARSE

MCSL benchmark suite provides traffic trace files for mapping of SPARSE application on mesh networks of different sizes. SPARSE is a medium size application with 96 tasks and 67 communication links. A communication link represents the flow of data from a task mapped on a source node to a task mapped on a destination node.

We simulated SPARSE on Noxim for simulation time of 100K cycles with the warm up period set to the initial 10K cycles. A 10x10 mesh NoC is selected with the buffer size of four and the packet size fixed to eight flits.

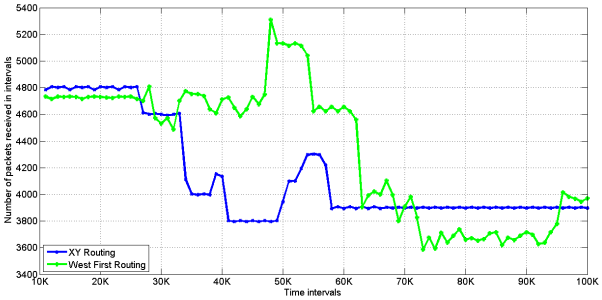

Figure 2: Number of received packets in the case of XY and west first routings for SPARSE application.

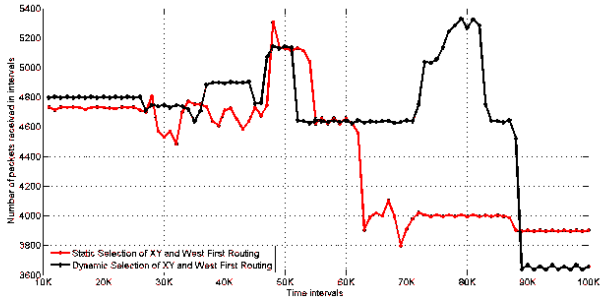

Figure 3: Number of received packets in the case of static and dynamic selection between XY and west first routings for SPARSE application.

Figure 2 shows the number of packets received after each interval, when XY and west first routings (i.e. two candidate algorithms) are applied on the SPARSE traffic trace file. In this plot, simulation period of 100K is divided into 100 intervals each with the length of 1000 cycles.

In Figure 2, we observe that for simulation period of 11K to 71K, number of received packets in the case of west first routing is more than or closer to that of XY routing algorithm. However, after 71K simulation cycles, number of received packets in the case of west first routing is significantly lower than that of XY routing. This can be explained by the nature of traffic pattern formed by the active flows after the time point of 71K cycles. The graph in Figure 2 suggests that the application of west first routing on the potential traffic pattern decreases number of received packets after the time point of 71K cycles, while XY routing performs better for that potential traffic pattern. Therefore we should apply west first algorithm from the start of simulation to 71K cycles and then switch to XY routing algorithm to enhance total number of received packets.

Figure 3 shows the number of received packets in the case of static and dynamic selection between XY and west first algorithms for SPARSE application. We observe that from the start of simulation to 71K cycles, number of received packets in the case of static selection is exactly the same as that for the west first routing (see Figure 2) but after 71K cycles, number of received packets is higher than that for both XY and west first routings.

To apply the dynamic selection between XY and west first algorithms, we first define CTPs for a node based on the congestion level on that node. In our setting of Noxim, each node has five FIFOs, each one with the length of four. Hence, the total length of five buffers is 20. We define CTPs as given below using the trial and error approach.

$$CTP = \{4, 20\}$$

$$AL = \{XY, \text{West First}\}$$

Figure 3 shows that the dynamic selection of two routing algorithms performs better as compared to the static selection. This is due to the fact that changes in the congestion level are monitored at the runtime and routing algorithms are selected accordingly.

Table 1 compares the throughput of individual application of XY and west first routing algorithms with the static and dynamic selection between them. Throughput is calculated as a ratio of the total number of packets received in the steady state phase to duration of the phase.

Table 1: Comparison of Throughput for XY, West First Routings and Static & Dynamic Selection Between Them for SPARSE Application.

Algorithm	Number of Received Packets in Steady State	Throughput
XY	39264	0.4908
West First	42032	0.5254
Static Selection	42539	0.5317
Dynamic Selection	46086	0.5760

We observe that the static selection of XY and west first algorithms improves throughput by 8.3% as compared to the XY routing and 1.2% as compared to the west first routing. For the dynamic selection case, throughput improves by 17.37% as compared to the XY routing and by 9.64% as compared to the west first routing.

5.2 Results for MPEG-4 Decoder

MPEG-4 decoder is another application for which we selected routing algorithms during its execution to improve throughput. We simulated the MPEG-4 decoder's task set on Noxim for a simulation time of 100K cycles with the warm up period set to the initial 10K cycles. A 4x4 mesh NoC is used with the buffer size of four and the packet size fixed to eight flits.

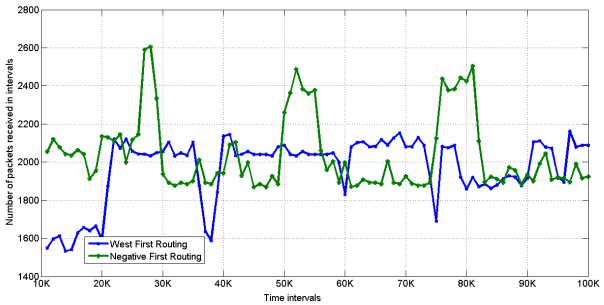

Figure 4: Number of received packets in the case of west first and negative first routings for MPEG-4 decoder application.

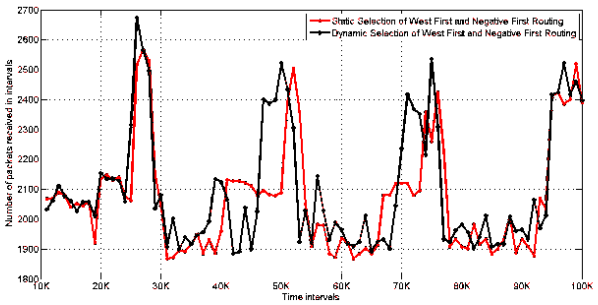

Figure 5: Number of received packets in the case of static and dynamic selection between west first and negative first routings for MPEG-4 decoder application.

Figure 4 shows the number of received packets when west first and negative first routing algorithms are applied individually on the MPEG-4 decoder application. It can be noted that the performance of these routing algorithms varies during simulation. For example, from the time point of 61K cycles to 74K cycles west first routing performs better than the negative first routing. On the other hand, after 74K simulation cycles, negative first routing exhibits better performance.

It suggests that we can switch the routing algorithm during execution of the application, to gain a higher throughput. This is demonstrated in Figure 5, where curves for number of received packets are drawn for the two cases i.e. static selection and dynamic selection between west first and negative first routings.

For the dynamic selection between two algorithms, CTPs and AL are defined as given below using the trial and error approach.

CTP = {4, 20}

AL = {West First, Negative First}

Table 2 compares the throughput of individual application of west first and negative first routing algorithms with the static and dynamic selection between them. Static selection between two routing strategies provides a throughput improvement of 5.39% and 1.74% as compared to the individual application of west first and negative first routing algorithms, respectively. Dynamic selection between two algorithms performs slightly better providing throughput increase of 6.74% and 3.05% as compared to west first and negative first routing algorithms, respectively.

6. CONCLUSION & FUTURE WORK

For real applications, throughput improves if we switch routing algorithms during execution of an application instead of running a single algorithm over the entire execution span. Next routing algorithm can be selected either in the static or dynamic way. Dynamic selection of algorithms performs better as compared to the static one because algorithms are selected at the node level instead of NoC level and the selection is driven by the congestion information.

In the future work, we shall investigate the switching overhead and the deadlock problem arising from the transient period when a node switches from one algorithm to another. Another interesting study can be investigating the effects of local algorithm selection on global congestion of the network. A possible option is dynamic selection of algorithms at the region level instead of the node level.

Table 2: Comparison of Throughput for West First, Negative First Routings and Static & Dynamic Selection Between Them for MPEG-4 Decoder Application.

Algorithm	Number of Received Packets in Steady State	Throughput
West First	22142	0.2460
Negative First	22935	0.2548
Static Selection	23335	0.2592
Dynamic Selection	23636	0.2626

7. REFERENCES

[1] D. Bertozzi, A. Jalabert, S. Murali, R. Tamhankar, S. Stergiou, L. Benini, and G. De Micheli. Noc synthesis flow for customized domain specific multiprocessor systems-on-chip. *Parallel and Distributed Systems, IEEE Transactions on*, 16(2):113–129, 2005.

[2] C. Bobda and A. Ahmadinia. Dynamic interconnection of reconfigurable modules on reconfigurable devices. *Design & Test of Computers, IEEE*, 22(5):443–451, 2005.

[3] L. Devaux, S. Ben Sassi, S. Pillement, D. Chillet, and D. Demigny. Draft: Flexible interconnection network for dynamically reconfigurable architectures. In *Field-Programmable Technology, 2009. FPT 2009. International Conference on*, pages 435–438. IEEE, 2009.

[4] F. Fazzino, M. Palesi, and D. Patti. Noxim-noc simulator. *Online] http://noxim. sourceforge. net.*

[5] C. J. Glass and L. M. Ni. The turn model for adaptive routing. In *ACM SIGARCH Computer Architecture News*, volume 20, pages 278–287. ACM, 1992.

[6] P. Gratz, B. Grot, and S. W. Keckler. Regional congestion awareness for load balance in networks-on-chip. In *High Performance Computer Architecture, 2008. HPCA 2008. IEEE 14th International Symposium on*, pages 203–214. IEEE, 2008.

[7] G. Haiyun. Survey of dynamically reconfigurable network-on-chip. In *2011 International Conference on Future Computer Sciences and Application*, pages 200–203. IEEE, 2011.

[8] J. Hu and R. Marculescu. Dyad: smart routing for networks-on-chip. In *Proceedings of the 41st annual Design Automation Conference*, pages 260–263. ACM, 2004.

[9] K. Lee, S.-J. Lee, and H.-J. Yoo. Low-power network-on-chip for high-performance soc design. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 14(2):148–160, 2006.

[10] W. Liu, J. Xu, X. Wu, Y. Ye, X. Wang, W. Zhang, M. Nikdast, and Z. Wang. A noc traffic suite based on

real applications. In *VLSI (ISVLSI), 2011 IEEE Computer Society Annual Symposium on*, pages 66–71. IEEE, 2011.

[11] T. Pionteck, R. Koch, and C. Albrecht. Applying partial reconfiguration to networks-on-chips. In *Field Programmable Logic and Applications, 2006. FPL'06. International Conference on*, pages 1–6. IEEE, 2006.