NoC Reconfiguration for CMP Virtualization

Francisco Triviño, Francisco J. Alfaro, José L. Sánchez
RAAP, Universidad de Castilla-La Mancha
Emails: {francisco.trivino, fco.alfaro, jose.sgarcia}@uclm.es

José Flich
DISCA, Universitat Politècnica de València
Email: jflich@disca.upv.es

***Abstract*—At NoC level, the traffic interferences can be drastically reduced by using virtualization mechanisms. An effective strategy to virtualize a NoC consists in dividing the network in different partitions, each one serving different applications and traffic flows. In this paper, we propose a NoC reconfiguration mechanism to support NoC virtualization under real scenarios Dynamic reassignment of network resources to different partitions is allowed in order to NoC dynamically adapts to application needs. Evaluation results show a good behavior of CMP virtualization.**

***Keywords*-Chip Multiprocessor; Network-on-Chip; Virtualization; Reconfiguration; Resource Management.**

I. Introduction

The Network-on-Chip (NoC) communication architecture has been proposed as a promising solution for providing high scalability in large CMP designs. Such NoCs are required to meet the challenges imposed by the most advanced chip technologies to become part of future Chip Multiprocessor (CMP) systems [1] consisting of a network of resources exchanging packets, while running different applications simultaneously.

CMP applications can be of diverse nature, making the traffic pattern completely unpredictable because of the very different program behaviors for different external inputs (e.g. computer vision, media processing, simulations, etc.). Taking into account the poor scalability degree that typical CMP applications exhibit, multiple applications with different requirements will run simultaneously in order to utilize the large number of cores.

In this scenario, all the resources in the CMP are shared by the applications. If this is not done in an efficient way, performance of any individual application can be seriously affected. At NoC level, the network interferences can be drastically reduced by using virtualization mechanisms. The keystone of the virtualization approach is that traffic from one application is not allowed to affect other applications.

To isolate the application traffic, we proposed the NoC resources virtualization and applied it to the CMP systems [2]. The proposal is based on the virtualization concept and allows to reduce execution time and network latency in a significant percentage. Specifically, we proposed a static scenario where four applications use the CMP at the same time. However, in a real system, applications enter and leave the system continuously in a dynamic way. In a dynamic scenario, the network must be reconfigured in order to allocate new partitions for new applications, In this paper, we propose a reconfiguration mechanism to deal with virtualized NoC under real scenarios. In these cases, the NoC should enable dynamic reassignment of network resources to different partitions in order to get adapted dynamically to application needs.

The remainder of this paper is organized as follows: in the following section, the related work is presented. In Section III we present the reconfiguration mechanism used to develop our NoC virtualization scheme. Section IV details the performance evaluation and shows the obtained results. Finally, Section V presents conclusions.

II. Related Work

If we focus on the CMP context, the virtualization model design is a multifaceted process involving several issues. For instance, virtualization involves the operating system and the different applications running together in the system. The memory system should be optimized for minimizing the interference among separate VMs to isolate better the single-workload of one application. To address this problem, *Michael R. et al.* proposed a variety of techniques in [3] focused on a CMP memory system for server consolidation. Another example can be found in [4] where authors address isolated cache usage, unmanaged shared caches and different cache partitions per application based on quality of service parameters. Unfortunately, in these studies the NoC is not taken into account.

In [5] the authors introduce the concept of the NoC virtualization and present some advantages based on the maximization of the network performance, the fault-tolerant capability, and the reduction in the power consumption. As a position paper, authors do not propose how to perform NoC virtualization and no evaluation is presented.

In [2], we proposed the use of the Logic-Based Distributed Routing (LBDR) mechanism [7] as an effective method of NoC partitioning at the routing level. We analyzed a static situation where four applications were running at the

This work was supported by the Spanish MEC and MICINN as well as European Comission FEDER funds, under Grants "Consolider Ingenio-2010 CSD2006-00046" and "TIN2009-14475-C04", respectively; it was also partly supported by Junta de Comunidades de Castilla-La Mancha under grants "PEII11-0229-2343" and "POII10-0289-724".

same time. In this scenario only four mapped applications are considered and they are settled at the beginning of the simulation until the end of the first application execution.

In this paper, we analyze real scenarios where multiple applications are executed dynamically. In this case, we develop a reconfiguration mechanism to deal with virtualized NoC in order to enable dynamic reassignment of network resources.

Other related works are focused on reconfigurables on-chip networks for CMPs. For instance, in [6] the authors provide an architecture that can implement different on-chip network topologies depending upon the application. Unfortunately, if the arrival of jobs is completely random, then the run-time required to fully utilize their architecture could be very complex. Our reconfiguration mechanism is based on a much simpler interconnection architecture where LBDR logic sets up the paths of the routing algorithm to maintain the illusion of a normal NoC.

III. Reconfiguration Mechanism

In this section we describe an effective, practical and fast method for reconfiguring the routing bits of the NoC which allows to virtualize the network resources (by dividing the NoC in different partitions) in a dynamic environment.

Firstly, note that shape and sizes of the CMP partitions are selected by the operating system resource manager which is independent of the reconfiguration mechanism. We only consider that the resource manager take into account the number of application threads (one thread requires one core).

If we focus on the network layer, our reconfiguration mechanism works assuming the use of the LBDR mechanism. The LBDR mechanism relies on two bit sets per output port at every switch of the mesh and a small logic block containing several gates. The first set lies in one bit per port and allows to define the connection pattern of the region. Each output port has a bit, referred to as *Cx* indicating whether a switch is connected through the *x* port. Thus, connectivity bits are *Cn*, *Ce*, *Cw*, and *Cs*. The second set lies in two bits per port and defines the set of turns that are allowed due to the applied routing algorithm. The bits for the east output port are labeled *Ren* and *Res*. They indicate whether packets routed through the east output port may take the north port or south port at the next switch, respectively. In other words, these bits indicate whether packets are allowed to change direction at the next switch. For north output port the bits are accordingly labeled *Rne* and *Rnw*, for west output port *Rwn* and *Rws*, and for south output port *Rse* and *Rsw*.

Figure 1 shows an example of the LBDR mechanism where the CMP with 16 nodes has been divided into two regions of 8 nodes. The LBDR bits are detailed for switch 6. The figure represents with arrows the routing restrictions (set of two consecutive links that cannot be taken by any packet). The routing bits mimic the routing restrictions. In the example, the SR [8] routing algorithm has been applied.

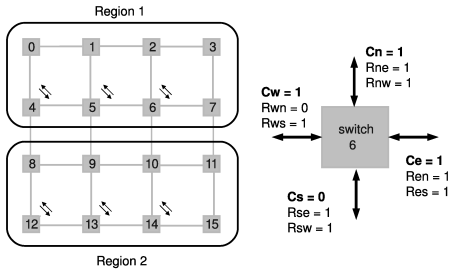

Figure 1. An example of the LBDR mechanism.

Assuming the previous routing configuration, and once the operating system starts allocating applications, we need to identify the new shapes that result by the creation of new partitions. For instance, Figure 2.(a) shows a situation where three applications have been allocated on the CMP. The LBDR bits have been adapted accordingly. First, connectivity bits establish the boundaries of the partitions (e.g. south connectivity bit of switches 2 to 7 are set to 0, and north connectivity bit of switches 6 to 11 are set to 0). Moreover, routing restrictions have been configured accordingly to prevent cycle dependencies on each partition separately (e.g. switch 5 has a bidirectional restriction in the west-north and north-west directions).

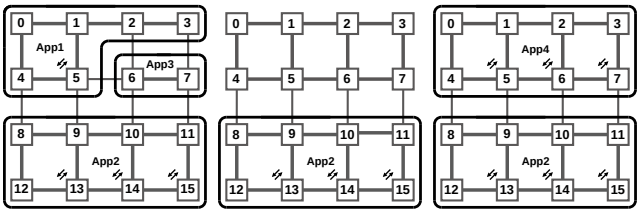

Figure 2. (a) Starting point, (b) App1 and App3 finish, and (c) App4 is settled.

In order to keep the normal NoC operation, when a new partition is created, the LBDR bits must be reviewed, and the LBDR configuration bits need to be updated, if necessary. Figure 2.(b) shows an example where starting from the situation of Figure 2.(a) both App1, and App3 applications have completed their execution. Then, a new application, App4, requests 8 nodes and the operating system selects the nodes 0 to 7 (Figure 2.(c)). In this situation, the LBDR bits for switches 0 to 7 must be reconfigured before launching App4. As can be deduced, the north connectivity bit of both 6 and 7 switches must be set to allow the communication with all nodes in the partition, and the south connectivity bit of switches 2 and 3. Finally, the routing restrictions must be computed and reconfigured for the new partition.

Note that this reconfiguration process relies on a static reconfiguration (no traffic on fly is involved) and affects only to parts of the network where no messages are traveling through the switches because the applications using them have already finished. This is very important because

otherwise deadlocks could appear and to avoid them either all the traffic of the CMP should be stopped before to make the reconfiguration, or a more complex reconfiguration mechanism should be considered. Thanks to the fact that there are no traffic interferences among different regions, we can perform local reconfiguration without affecting the rest of the regions of the CMP. In the previous example, only the LBDR bits of the free nodes 0 to 7 must be reconfigured. For this reason, the reconfiguration mechanism ensures a deadlock-free situation.

IV. Performance Evaluation

In this section, we evaluate by simulation the behavior of the virtualization scheme considering the reconfiguration mechanism previously described.

The simulation platform is based on the Simics-GEMS simulators [14]. The NoC is modeled with a cycle-accurate flit-level network simulator. The network simulator models the LBDR mechanism, the SR routing algorithm, and the proposed reconfiguration mechanism.

Regarding the CMP model, we model a homogeneous 16-CMP with in-order cores. We include the simulation parameters modeling the complete CMP architecture in Table I. As workload we have used applications from both SPLASH-2 [11] and PARSEC v2.1 [12] benchmark suites.

Table I
Overview of the CMP configuration.

Parameter	Configuration
CPU / OS	16 x UltraSparc III processor / Solaris 10
L1 cache	Instructions & Data, 64 bytes block size, 2-way associativity, 256 banks, 32 Kbytes per core, and 2 cycles latency
L2 cache	64 bytes block size, 4-way associativity 512 banks, 1 Mbyte/core, 10 cycles latency
Main memory	2 Gbytes total, 250 cycles latency 3D-Stacking [9]
Coherence protocol	MOESI-CMP-Directory & First-Touch [10]
Topology	2D-mesh regular 4x4
Router	4 flits buffer queue size, 8 bytes flit size SR routing algorithm
Network frequency	The NoC works at the same frequency as the processors

A. Scenarios

In this study we have considered different scenarios. In each scenario we run 5 sets of applications. Each set contains 20 applications with different inputs. The application requirements are only based on the number of cores needed to run the applications such as we have considered that each thread must be run in a different core. These application requirements are configured ranging from 2 to 8 threads. The specific applications in each set and the number of threads for each application are randomly selected. The resource manager automatically assigns the CMP resources to the applications in a sequential way. The applications are stored in a FIFO queue until the resource manager has enough resources to allocate them in the CMP. In our opinion, this covers a wide range of different situations.

The first scenario we have considered is a baseline NoC with no mechanism to isolate the traffic of applications and, thus, no reconfiguration mechanism is needed. In the baseline scenario (labeled in the figure as BL) each application uses the entire NoC where the threads of each application will be distributed among the nodes through a random distribution. When one application finishes the resources it used will be free and other applications in the waiting list could be assigned to these resources. When there are enough resources to allocate other application the system starts the next application execution and so until all the applications of the working set finish. In this baseline scenario, the traffic in the NoC generated by an application is affected by the traffic generated by the other applications because all of them share and use the whole network.

In the second scenario we enable the partitioning mechanism to create partitions. In this scenario, as a result of the partitioning process, messages generated by an application running in a given partition can only use the network resources belonging to that partition (we called this scenario “Virtual-Regions” (VR)). We assume the same CMP configuration than in the BL scenario but, in this case, the applications are assigned to virtual regions. As the traffic of the applications must be isolated, the resources should be assigned in a contiguous way. For this reason, we use a contiguous strategy [13] from the state-of-the-art in order to assign resources to the applications. Note that the resource manager is independent of the reconfiguration mechanism. In this scenario each application is assigned to one virtual region in a similar way that in the example of Figure 2. In this scheme, traffic from one region is not allowed to traverse other regions. Notice that the partitions will have different sizes in order to satisfy the application requirements (number of application threads). Note that the LBDR mechanism is able to support irregular partitions, and as a consequence, we can maximize the system utilization.

Finally, we have considered an additional scenario in order to highlight the negative effect of the traffic interferences. In this scenario, the applications are mapped in the same way as for the VR scenarios using the same contiguous allocation strategy. However, in this case the messages belonging to one domain can cross the boundaries of its domain to reach their destinations because the NoC is not virtualized (we called this scenario “Virtual Domains” (VD)). The VD scheme will permit us to study the importance of just the NoC virtualization because in the VR cases the traffic is completely isolated while in the VD cases traffic of different applications can compete for the same network resources. Finally, in the VD scheme we do not apply any reconfiguration mechanism because it is not necessary, on the contrary, we use a SR routing for the complete mesh.

Note that only the VR scenario needs our NoC reconfiguration process. In this case, each time this algorithm has

been applied, we consider an additional execution time that depends on the SR algorithm cost [8] before the application can start its execution. Regarding the threads mapping strategy, we have not made any time difference in applying the contiguous strategy and we consider the same time to map the threads regardless the followed strategy.

B. Experimental Results

In this section, we show the experimental results obtained in the evaluation process. As in the BL case the threads are randomly distributed through the cores, each BL value shown in the figures is the average of 30 different simulations where the confidence interval has been set to 95%.

Figure 3 shows the total execution time for the 5 sets of applications (x-axis). The y-axis depicts the difference in total execution time among the BL, VR and VD scenarios. Results are shown in normalized terms, with respect to the execution time of the BL case.

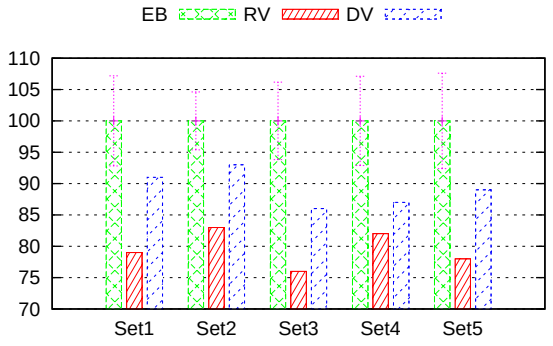

Figure 3. Normalized total execution time of the different scenarios.

As can be seen, results show how the interferences among the different application traffics can affect significantly the total execution time. In the BL case, all the applications must compete for the network resources as all of them are spread over the CMP, sharing so the network. Consequently the interferences affect all the applications. When multiple applications share the network, the communication cost due to the interferences among messages affects the performance of individual applications.

On the other hand, when we use the VD scenario there are less traffic interferences than in the BL scenario and so the performance is much better and stable. For instance, the total execution time for the VD case is decreased by 14% approximately in the Set3 when compared with the BL case. If we compare the VR and the VD scenarios, although having the same resource assignment, further benefits are obtained for the VR scenario, as total execution time decreases to 10% when compared with the VD scenario (in the Set3).

V. Conclusions

In [2], we presented a NoC virtualization mechanism to isolate the application traffic. In this paper we include the reconfiguration mechanism in order to adapt the NoC to the application needs. We show that the VR scenario can lead to important performance improvement. In particular, we have shown that the VR scenario reduces (for all the application sets that we have tested) from 5% to 12%, approximately, the total execution time compared with the VD scenario (where no isolation is provided and no reconfiguration is needed). Also both the network latency and the link load follow the same trend that the execution time. This fact is due to the elimination of the interferences among messages belonging to different applications, because in both cases the rest of the CMP (mainly cores and caches) is equally partitioned.

References

[1] F. Gilabert, et al., *Designing Network On-Chip Architectures in the Nanoscale Era*, 2010.

[2] F. Triviño, J. L. Sánchez, F. J. Alfaro, and J. Flich, "Virtualizing network-on-chip resources in chip-multiprocessors," *Microprocessors and Microsystems*, 2011

[3] M. R. Marty and M. D. Hill, "Virtual hierarchies to support server consolidation," *ISCA*, 2007.

[4] F. Guo, H. Kannan, et al., "From Chaos to QoS: Case Studies in CMP Resource Management," *SIGARCH Comput. Archit. News*, 2007.

[5] J. Flich, et al., "On the Potential of NoC Virtualization for Multicore Chips," *MuCoCoS*, 2008.

[6] M. M. Kim, et al., "Polymorphic on-chip networks," *ISCA*, 2008.

[7] J. Flich and J. Duato, "Logic-Based Distributed Routing for NoCs," *IEEE Comput. Archit. Lett.*, 2008.

[8] A. Mejia, et al., "On the Potentials of Segment-Based Routing for NoCs," *ICPP*, 2008.

[9] B. Black, et al., "Die Stacking (3D) Microarchitecture," *MICRO*, 2006.

[10] S. Cho and L. Jin, "Managing Distributed, Shared L2 Caches through OS-Level Page Allocation," *MICRO*, 2006.

[11] S. C. Woo, et al., "The SPLASH-2 programs: characterization and methodological considerations," *ISCA*, 1995.

[12] C. Bienia and K. Li, "PARSEC 2.0: A New Benchmark Suite for Chip-Multiprocessors," *MoBS*, 2009.

[13] A. G. Solheim, et al., "Routing-contained virtualization based on Up*/Down* forwarding," *HiPC*, 2007.

[14] F. Triviño, F. J. Andujar, F. J. Alfaro, J. L. Sánchez, and A. Ros, "Self-Related Traces: An Alternative to Full-System Simulation for NoCs," *HPCS*, 2011.

[15] A. Mejia, "Design and Implementation of Efficient Topology Agnostic Routing Algorithms for Interconnection Networks", PhD dissertation, University of Valencia, 2008.