Journal of Systems Architecture 58 (2012) 126–139

Contents lists available at SciVerse ScienceDirect

Journal of Systems Architecture

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

Network-on-Chip virtualization in Chip-Multiprocessor Systems

Francisco Triviño [a,*], José L. Sánchez [a], Francisco J. Alfaro [a], José Flich [b]

[a] Department of Computing Systems, Universidad de Castilla-La Mancha, Albacete, Spain
[b] Parallel Architectures Group, Universitat Politècnica de València, Valencia, Spain

ARTICLE INFO

Article history:
Received 6 May 2011
Received in revised form 22 January 2012
Accepted 5 March 2012
Available online 17 March 2012

Keywords:
Chip Multiprocessor
Network-on-Chip
Virtualization

ABSTRACT

It is expected that Chip Multiprocessor Systems (CMPs) will contain more and more cores in every new generation. However, applications for these systems do not scale at the same pace. In order to obtain a good CMP utilization several applications will need to coexist in the system and in those cases virtualization of the CMP system will become mandatory. In this paper we analyze two virtualization strategies at NoC-level aiming to isolate the traffic generated by each application to reduce or even eliminate interferences among messages belonging to different applications. The first model handles most interferences among messages with a virtual-channels (VCs) implementation reducing both execution time and network latency. However, using VCs results in area and power overhead due to the cost of control and buffer implementation. In contrast, the second model is based on the resource partitioning strategies which results in a space partitioning of the CMP chip in several regions. For this last model, Virtual-Regions (VR), we use a reconfiguration algorithm of the network that is able to dynamically adapt the network partitions in order to satisfy the application requirements. The paper shows a comparison of both models and identifies their main advantages and disadvantages. From our experimental results, we show that our proposal obtains in terms of execution time average improvements of 30% for parallel applications when compared to a baseline scenario. Moreover, when compared to a VCs implementation, our proposal improves the average execution time by 9% for parallel applications.

1. Introduction

In every new generation, Chip Multiprocessor Systems (CMPs) contain more and more cores in the same chip [1,2]. Although these cores usually do not run as fast as the highest performing single-core processor, they all improve the overall performance and exhibit a good performance/power trade-off. Therefore, it is quite likely that in some years systems with hundreds or even thousands of cores will be in the market.

Performance of those CMP platforms depends not only on the number of cores but also heavily on platform resources such as cache, external memory, as well as the interconnection of all the CMP components.

In CMPs efficient communication among nodes is an important requirement. Networks on chip (NoCs) [3] are required to meet the challenges imposed by the most advanced chip technologies to become part of future CMP systems.

Traditionally, the CMP platform is well designed for a single parallel application. In that case there are no cache interferences or network collisions. However, most applications do not present a good scalability degree. For instance, in [4] the authors provide a study of the scalability on the PARSEC v2.1 benchmark suite [5] assuming all the components in the CMP architecture. As a result of this study, the authors highlight that almost all the applications scale well up to 16 cores, but they fail to scale from 16 cores upwards.

Therefore, one way to take full advantage of future manycore CMPs is to allow several applications running simultaneously. However, this must be done with care. In such scenario, the CMP load and the interferences among applications will increase. For instance, the network latency in CMPs greatly increases as the number of applications running together increases [6], and so the NoC has a large impact on the applications performance (final execution time). This is true even for multithreaded cores and especially for applications generating high cache miss rates. Then, minimizing network latency should be a priority on interconnects for such systems.

Therefore, in this scenario where several applications run simultaneously on a CMP, all the resources in the CMP are shared by the applications, and if resources usage is not efficient, performance of any individual application can be seriously affected. Note that in those cases we do not want to improve the performance of individual applications, but to keep the individual application performance in the same values that when they are run alone,

* Corresponding author.
E-mail addresses: ftrivino@dsi.uclm.es (F. Triviño), jsanchez@dsi.uclm.es (J.L. Sánchez), falfaro@dsi.uclm.es (F.J. Alfaro), jflich@disca.upv.es (J. Flich).

1383-7621/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.sysarc.2012.03.001

maximizing thus the overall CMP performance by taking full advantage of the available resources.

In this paper we address this problem using the virtualization concept. In this way, applications obtain a subset of the CMP resources. CMP virtualization studies usually are only focused on the cores or the memory system [7,8] but do not address on-chip network virtualization.

In previous works, we addressed some of these problems. In [6] we presented a NoC-level virtualization approach (called Virtual Regions, VR) that splits CMP resources into several regions devoting each region to a single application. Fig. 1 shows the case where three applications are mapped to the CMP following our VR scheme. Each such processing element is attached to a local router which connects the core to the neighboring nodes via a NoC. However, in that work we made some simplified assumptions, in order to quickly analyze the behavior of the network. The results showed that the CMP performance could actually be greatly improved if a virtualization technique was used. However, this method presents some limitations as the need of a reconfiguration strategy at operating system level, the fragmentation of the available resources, etc. An alternative to decouple traffic from different applications is the use of virtual-channels (VCs). Therefore, it is worthy to compare the VR approach against the VCs where traffic generated by different applications is injected through different VCs. Indeed, VCs can be seen as a way to implement virtual networks (we call them virtual lanes, VLs). If there are more applications than VLs, several applications could be allocated into the same VL. In VR it is also possible to run simultaneously more applications than the number of partitions in the CMP. In that case, several applications would be allocated in the same partition. Note also that only applications into the same partition would interfere among them.

Of course, these approaches involve several CMP components and different actions must be performed. The operating system in close cooperation with a hypervisor must take into account the requirements of each application. Based on these requirements and the set of available resources it must decide if the application can be mapped and run. If possible, depending on the approach used, a VL (a new one or a shared one) is assigned to the application taking into account the VCs approach, or a partition (a new one or a shared one) is assigned to the application, if space partitioning is considered. In addition, for this last approach, the system should enable dynamic reassignment of cores/caches and network resources to different partitions.

This paper presents, in an unified manner, a virtualization platform that includes a reconfiguration mechanism. Unlike the previous version presented in [6], this new space partitioning takes into account not only a complete evaluation environment where different applications with very different requirements enter and leave the system in a dynamic environment, but also the resource allocation strategy in order to allocate the threads of the applications in a contiguous way. In particular, the new virtualization platform uses a reconfiguration mechanism of the Logic-Based Distributed Routing mechanism in order to allow the dynamic creation of partitions in the CMP. In this case, we use a contiguous allocation strategy for scheduling the parallel applications. We also present the evaluation of the proposed technique using real applications under realistic conditions. We also explore the trade-offs in varying the number of shared memory controllers. Indeed, as the number of cores increases, and due to the pin count limitation, the ratio between number of cores and memory controllers will increase. Finally, we have included a VCs virtualization model in order to explore a common alternative for NoC virtualization. This approach is fully analyzed and different versions of this approach are evaluated.

Summing up, the approaches presented here (VR and VCs) try to improve the applications performance reducing as much as possible the negative effect of network traffic interferences. Both are based on the virtualization concept and their aim is to provide traffic isolation when running simultaneously several applications. To carry out our study, we model a detailed NoC, and show the performance of several application sets with different virtualization schemes using a full system simulator.

The structure of this paper is as follows: Section 2 presents the related work. In Sections 3 and 4 we present our proposals for isolating the traffic of applications at NoC level. Section 5 details the performance evaluation and Section 6 shows the obtained results. Finally, Section 7 presents conclusions and directions for future work.

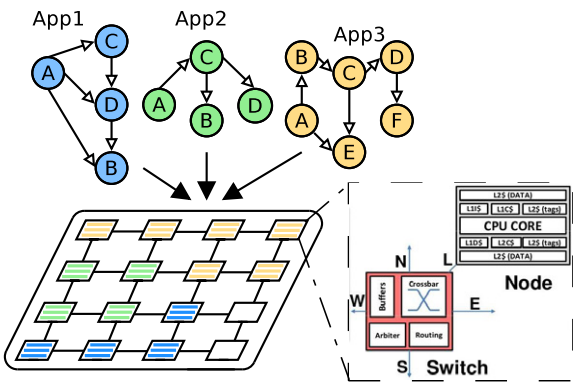

Fig. 1. Generic VR NoC architecture, applications, and its mapping to the NoC are shown. Each node consists of an on-chip router, buffers, processing element, and system memory.

2. Related work

Virtualization has been applied to different domains like virtual machines (VMs), virtual memory and virtual servers in data centers. In these scenarios, multiple server applications are deployed onto VMs, which then run on a single, more-powerful server. Many different workloads are consolidated together and hardware performance isolation [9] becomes a desired feature for running applications with different priorities identified by the user or system administrator.

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, *Marty and Hill* proposed a variety of techniques in [7] focused on a CMP memory system for server consolidation. Another example can be found in [8] where authors address isolated cache usage, unmanaged shared caches and different cache partitions per application based on quality of service parameters.

Regarding the interconnect system, VCs are often proposed to isolate several classes of traffic in a network [10–13]. The physical channel is shared by several VCs with independent buffer queues. In this way, VCs are a partial solution to virtualize the NoC as they isolate traffic from different applications but they share the physical channels. Moreover, using VCs in the NoC context has also important drawbacks. Their implementation results in an area and power overhead due to the cost of control and buffer implementation. In addition, if more applications are considered the number of necessary VCs increases. Moreover, if we want to guarantee performance bounds to applications, VCs are not enough. On the other hand, VCs are also used to avoid protocol-level deadlocks thus leading to a system where the number of required VCs multiplies.