Microprocessors and Microsystems xxx (2010) xxx–xxx

Contents lists available at ScienceDirect

Microprocessors and Microsystems

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

Virtualizing network-on-chip resources in chip-multiprocessors

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

[a] Department of Computing Systems, University of Castilla-La Mancha, Albacete, Spain
[b] Parallel Architectures Group, Technical University of Valencia, Valencia, Spain

ARTICLE INFO

Article history:
Available online xxxx

Keywords:
Network-on-chip
Virtualization
Performance evaluation

ABSTRACT

The number of cores on a single silicon chip is rapidly growing and chips containing tens or even hundreds of identical cores are expected in the future. To take advantage of multicore chips, multiple applications will run simultaneously. As a consequence, the traffic interferences between applications increases and the performance of individual applications can be seriously affected.

In this paper, we improve the individual application performance when several applications are simultaneously running. This proposal is based on the virtualization concept and allows us to reduce execution time and network latency in a significant percentage.

1. Introduction

In order to continue increasing the computing speed, current semiconductor manufacturing techniques include multiple cores on the same chip. The microprocessor industry has chosen multicore chips instead of an increasingly powerful uniprocessor [1]. Power usage, heat generation or cost are some of the main reasons against the latter motivating that decision. Although these cores do not necessarily run as fast as the highest performing single-core processors, they all improve the overall performance. Therefore, chips containing tens or even hundreds of identical cores are expected in the future. Chip multiprocessors (CMPs) are an excellent example of these systems [2–6].

To take full advantage of CMPs, it is also expected that several applications will run simultaneously on such CMP systems. Moreover, as the number of cores is increased, it is expected that the number of applications to run in the same CMP also increases. All those 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, animation, simulations, data mining, etc.). As a consequence, the CMP load will also increase, which may affect the performance of individual applications.

Fig. 1 shows results about several performance metrics both when there exists only one application[1] running in the system, and when it shares network resources with others which are represented by stress load in this test. In both cases the applications are running with 16 threads where each one is mapped in one core of a 4×4 mesh CMP. We have chosen synthetic traffic set up to 0.3 packet injection rate to represent the stress load. Finally, note that the results are shown in normalized terms compared with the performance metrics when the application runs alone.

In the figures the execution time increases 37% for the Blackscholes application and 25% for the Streamcluster, approximately. When we apply the stress traffic in the network, not only the execution time is affected. As we can see, the stress traffic has also a direct impact on whole CMP resources. For instance, the total application cache misses are increased by 31% for the Blackscholes and 21% for the Streamcluster application when both are running in shared mode.

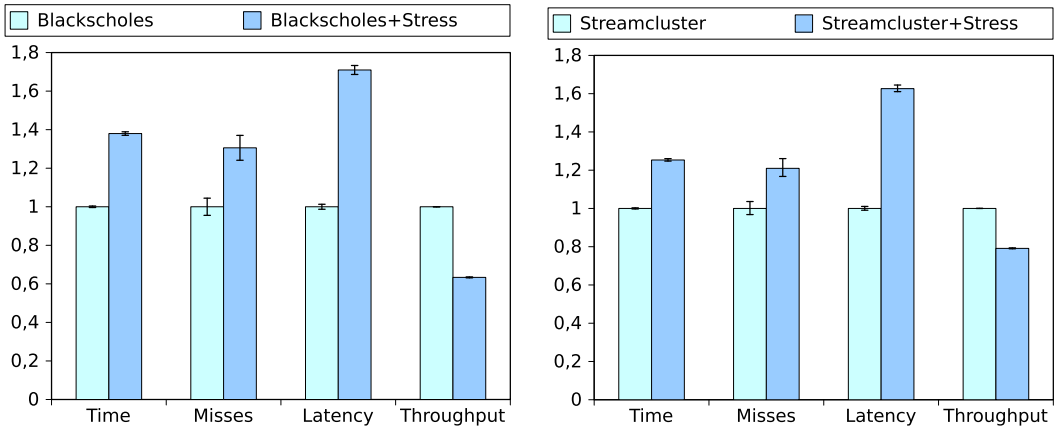
Therefore, it is necessary to improve the individual application performance when several applications are simultaneously running in a CMP system. In this scenario, all resources in the CMP are shared for the applications. If this is not done in an efficient way, performance of any individual application can be seriously affected. The work presented here highlights this problem and motivates the performance differentiation desired for the applications.

Our proposal to avoid these problems is based on isolating the traffic of each application so that the CMP could guarantee the performance requirements that applications need. This results in a partitioning of the CMP in several regions, each one composed of a subset of available resources in the CMP system. Fig. 2 shows an example of applying the virtualization mechanism to a CMP, where four partitions have been created which only share off-chip memory. Notice that the size of the partitions can be performed in order to satisfy the application's requirements and, therefore, it is possible to assign applications to partitions with different size. In this example, the CMP has been partitioned to allow four applications to run simultaneously with each one allocated in one different region not having interaction among them. Note that in

* 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).

[1] We have chosen two applications (Blackscholes and Streamcluster) selected from the recently released PARSEC v2.1 benchmark suite.

0141-9331/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.micpro.2010.10.001



(a) Blackscholes application. (b) Streamcluster application.

Fig. 1. Effect of stress traffic on two PARSEC applications.

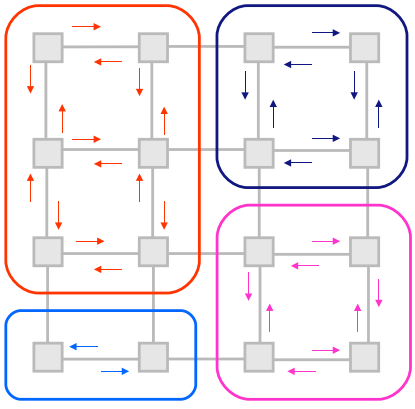


Fig. 2. Example of the virtualization mechanism.

this scenario it is also possible that more applications than four run simultaneously in the CMP. Then, several applications would be allocated to the same region. Only applications belonging to the same region could interfere among them.

Of course, this technique involves several CMP components and different actions must be performed. The operating system in close cooperation with a hypervisor must analyze the requirements of each application. Based on these requirements and the set of available resources it must be decided if the application can be run. If possible, a new partition is created for the application or the application is included in an existing one. The system should enable dynamic reassignment of cores, caches and memory to different regions.

Specifically, in this paper we focus on one of these components: the on-chip network. In CMPs, a requirement for a high-performance on-chip interconnect emerges allowing efficient communication among cores. Networks-on-chip (NoCs) can reduce the transmission delays to acceptably low values and allow an efficient communication among cores, cache levels and memory controllers. These NoCs are required to meet the challenges imposed by the most advanced chip technologies to become part of future CMP systems [7–12].

This CMP component has a major impact in performance and is responsible for much of the miss-latency in all our experiments. Moreover, we have observed in CMPs that the latency greatly increases as the number of applications running together increases, and so the NoC has a large impact on the applications performance (final execution time). This is true even for multithreaded cores and applications with large miss rates. The performance of applications is mainly constrained by latency and, therefore, minimizing latency should be a priority on interconnects for such CMPs.

In this paper, we propose, discuss and evaluate to isolate the traffic of different applications in order to reduce as much as possible the negative effect of the traffic interferences on the applications performance. Our proposal is based on the effective use of the LBDR bits as a means to virtualize the NoC. We propose two ways of isolating the traffic of different applications in order to reduce or even to eliminate the traffic interferences. We show that enabling and enforcing a virtualization mechanism is much more effective for managing the resources than a baseline NoC scheme.

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

2. Related work

As we have already shown in Section 1, if traffic isolation is not enforced by the NoC, the traffic interferences have negative effects in the performance of the applications. For this reason, the traffic isolation is the main property a NoC system must achieve in order to minimize interference among the different applications. In such a situation, traffic from one application is not allowed to affect other applications.

On the other hand, virtualization offers an opportunity for improving the applications performance in the context of multicore systems. The recent re-emergence of virtualization has been applied to different domains like virtual machines, virtual memory, and virtual servers in data centers. In these scenarios, multiple server applications are deployed onto virtual machines (VMs), which then run on a single, more-powerful server. Many disparate workloads are consolidated together and hardware performance isolation [13] becomes a desired feature for running applications with different priorities identified by the user or system administrator. On clusters, several studies [14–17] have focused on performance isolation by contiguous node allocation. The study presented in [14] focus on job scheduling techniques in conjunction with a contiguous node allocation that improves the run-time jobs (due to reduced link contention and lower communication overhead). All the above studies focus exclusively on allocating parallel jobs on clusters, but do not address the issue of resource management onto the CMP systems.

Please cite this article in press as: F. Triviño et al., Virtualizing network-on-chip resources in chip-multiprocessors, Microprocess. Microsyst. (2010), doi:10.1016/j.micpro.2010.10.001

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 et al. proposed a variety of techniques [18] focused on a CMP memory system for server consolidation. Another example can be found in [19] where the authors address isolated cache usage, unmanaged shared caches and different cache partitions per application based on quality of service parameters.

Regarding the interconnect system, in [20] the authors introduce the concept of the NoC virtualization and present some advantages. For instance, they claim that due to the low parallelism level that the current applications provide, the virtualization of a NoC provides a partitioning mechanism that allows to manage the resources and assign them to the applications in an efficient way. Another advantage relies on the yield factor. In a virtualized scheme failed components can be excluded by a smart partitioning scheme that logically divides the NoC into different regions. Similarly, cores and interconnect components that are otherwise idling can be put to sleep mode and excluded at the NoC level thus reducing power-consumption. As a position paper, authors do not explain how to perform NoC virtualization and no evaluation is presented.

Virtual channels (VCs) are often proposed to isolate several classes of traffic in the network [21–24,12]. 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 channel.

However, 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 traffic classes are considered the number of necessary VCs increases. Moreover, if we want to guarantee performance bounds to applications, VCs are not enough. Indeed VCs are also used to ensure such guarantees, as well to avoid protocol-level deadlocks thus leading to a system where the number of required VCs explodes.

Moreover, if we want to offer different treatment for each application, VCs are not enough. A connection admission control, and virtual output queuing at least at switch level are needed to guarantee certain performance levels.

Summing up, a virtualized NoC can improve the CMP performance when several applications are running together. A virtualized NoC may be viewed as a network that partitions itself into different regions, with each region serving different applications and traffic flows without interferences among applications of different regions. The virtualization concept requires that paths need to be adapted. As packets are not allowed to cross application boundaries, the routing algorithm must be adapted. Recently, the Logic-Based Distributed Routing mechanism (LBDR [25]), which is an effective method allowing such partitioning at the routing level, has been proposed. With this mechanism, and using as few bits per router as 8, the network can be reconfigured in an effective way. Moreover, VCs are not necessary thereby saving area and power. However, the way to use those bits for virtualization purposes has not been defined.

3. NoC virtualization

Generally, a CMP system consists of homogeneous nodes (tiles) where each one contains a processing element, cache memory and the local router which connects the node to the neighboring nodes building thus the NoC. One packet, when generated in the local processor, is sent to the router via a network interface. Then, the packet moves to the next router on its path depending on the routing algorithm, and the process is repeated until the packet arrives to its destination.

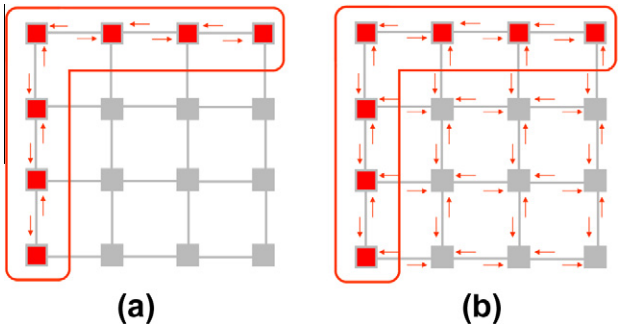


Fig. 3. Communication flows for (a) VR and (b) VD models.

In this paper we focus on the NoC presenting a technique to isolate the traffic of different applications in order to reduce as much as possible traffic interferences. In this case, applications do not share cores, caches nor network resources.

When the virtualization mechanism is used several partitions need to be configured in the CMP. We have, at the network level, two options. In the first case, 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. In the second case, on the contrary, it is also possible that those messages are allowed to use NoC resources in other partitions. The first option enforces partitioning at the network level, while the second enforces partitioning only at node level.

In order to distinguish these two possibilities we introduce the following definitions:

- *Virtual-regions* (VR): Virtual regions are partitions which are composed of a subset of contiguous nodes[2] and the links that interconnect them. These nodes are devoted exclusively to an application or group of applications sharing these resources. We group all the resources involved in a virtual region. In this scheme, traffic from one region is not allowed to traverse other regions.
- *Virtual-domains (VD):* Virtual domains are also composed of a subset of contiguous nodes but, in this case, messages generated by nodes in a partition can use links (and routers) of the NoC belonging to other partitions in order to reach their destinations. As in the previous case, these nodes are also devoted to an application or group of applications that share these resources. Therefore, this case is equal to the previous one but without the restriction that messages cannot cross the boundaries of the region.

Fig. 3 shows the differences between the VR and VD cases. As seen in (a), messages generated by nodes in the region (represented by arrows) only use links that are within the region. In this particular case and under high rates of packet transmission, links belonging to the virtual region could be heavily congested.

On the other hand, in the virtual domain case (b), nodes can communicate via all the links in the NoC and thus distribute the network load. However, in this case the traffic generated in a given

[2] A node is composed by a local core, a private L1 cache, a shared L2 cache physically distributed among the nodes of the whole CMP, and a router connecting to neighbor nodes.

Please cite this article in press as: F. Triviño et al., Virtualizing network-on-chip resources in chip-multiprocessors, Microprocess. Microsyst. (2010), doi:10.1016/j.micpro.2010.10.001

partition is not isolated and can interfere with other traffic generated in other partitions. On the contrary, in the virtual-regions case, all traffic is fully isolated and the communication overhead due to interferences of messages belonging to different regions is avoided.

Note that the specific communication paths depend on the routing algorithm used in the network. The routing algorithm is the only architectural modification needed to support these virtualization models at NoC level. The algorithm should be flexible enough to allow irregularly shaped regions and should be designed taking into account the tight constraints applied to multicore chips regarding latency, power-consumption, and area. Recent solutions such as the LBDR mechanism [26] enable the virtualization concept in a NoC and allow the allocation of partitions in a mesh topology with small hardware requirements.

The Logic-Based Distributed Routing (LBDR) mechanism relies on two sets of bits per output port at every switch 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 output port north the bits are accordingly labeled *Rne* and *Rnw*, for output port west *Rwn* and *Rws*, and for output port south *Rse* and *Rsw*.

Fig. 4 shows an example of the LBDR mechanism where the connectivity bits and the routing bits are detailed for switch 6. The CMP with 16 nodes has been divided into two regions, each one with eight nodes.

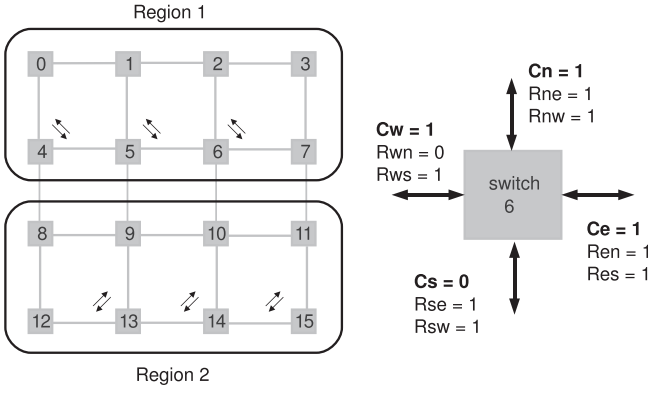


Fig. 4. An example of the LBDR mechanism.

On the other hand, we would like to remark that partitions with different sizes and shapes are supported by this scheme without changes in the hardware. This fact offers a great flexibility to the application scheduler when assigning free resources in the CMP. Fig. 5 shows different CMP configurations where the partitions have different sizes. In these cases, we need to carefully compute the LBDR bits within the network since deadlock and unconnected nodes may arise. Fig. 5a shows the case where three partitions have been set. In partitions P1 and P3 the up*/down* [27] routing algorithm is used while in P2 the configuration is obvious (only connectivity bits must to be configured). For this example, there are no differences between VR and VD cases because all the partitions are regular, and the messages have no opportunity to cross the boundaries of the different partitions.

In Fig. 5b–d other examples are provided. Notice that the partitions have also different sizes. These figures represent examples of partitions with irregular shapes fully supported by LBDR. Indeed, the LBDR mechanism is able to support irregular partitions, and as a consequence, we can obtain a maximization of the system utilization. For these specific cases, there are differences between VR and VD cases due to the irregular partitions.

To sum up, our proposal is based on the effective use of the LBDR bits as a means to virtualize the NoC. This fact allows us to manage different partition sizes at same time without changing the hardware. We consider two possibilities (configuration of routing bits and connectivity bits): in the VR model the messages can only travel through specific regions while in the VD model they can cross the boundaries.

4. Performance evaluation

In this section, we evaluate the behavior of our proposal using simulation. We describe the simulation environment (based on a full-system), the integration of the different tools to obtain the full-system simulator, the benchmarks used in our experiments as well as our experimental setup, followed by a detailed discussion of the results obtained.

4.1. Simulation environment

As it is well known, there is a combined effect among all the parts of a CMP architecture that may impact the obtained results when a particular component is studied. The simulation environment we use consists in a full-system simulator that allows us to simulate realistic workloads instead of synthetic traffic thus taking into account all the interactions.

Fig. 6 shows a global scheme of all the simulation tools running together. Our full-system simulator is based on Simics–GEMS. Simics [28] simulates different processors at the instruction-set level (the top of the figure). The General Execution-driven Multiprocessor Simulator (GEMS) [29] is a simulation toolset to evaluate

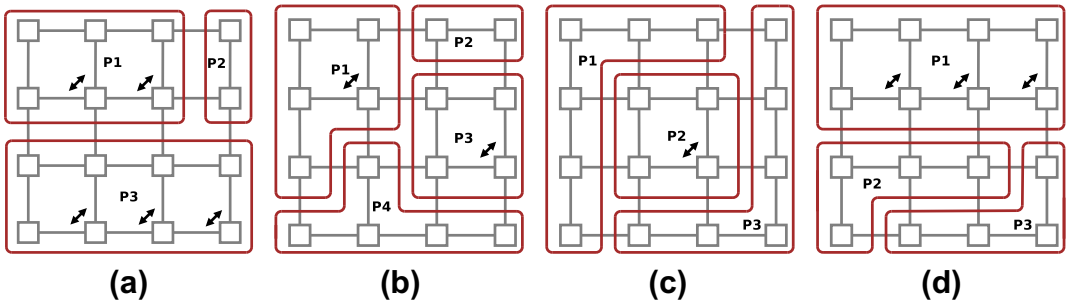


Fig. 5. Four possible cases fully supported by LBDR.

Please cite this article in press as: F. Triviño et al., Virtualizing network-on-chip resources in chip-multiprocessors, Microprocess. Microsyst. (2010), doi:10.1016/j.micpro.2010.10.001

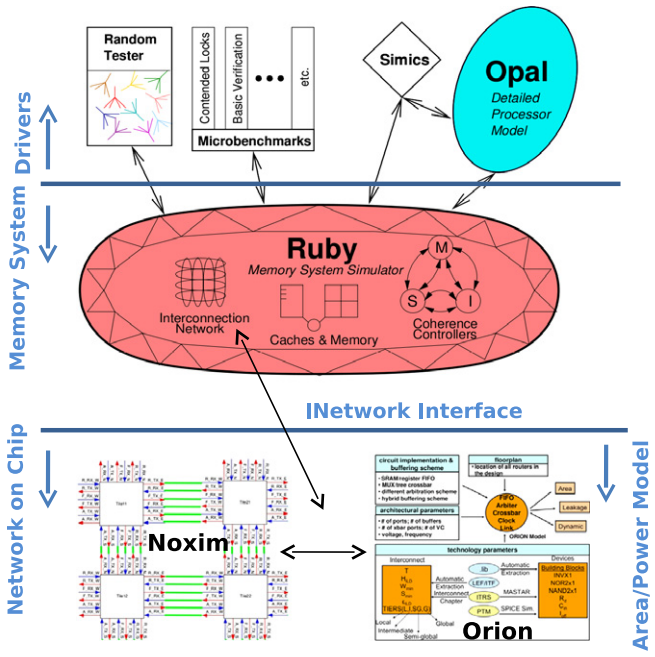


Fig. 6. Global view of the simulation tool.

multiprocessor architectures using Simics. GEMS is composed of two main simulation modules: Ruby and Opal. Ruby (in the middle of the figure) is a simulator of a CMP memory system that models caches, cache controllers, system interconnect, memory controllers, and banks of external memory. Ruby can also use coherence protocols, designed to preserve the integrity of data distributed on different levels of the memory hierarchy. Opal allows to simulate an out-of-order processor.

The interconnection network allows to communicate among cache, memory controllers and processors. We have replaced the interconnection model included in Ruby with other NoC simulator called Noxim [30] (represented on the bottom of the figure). The Noxim kernel is based on SystemC and simulates a NoC in detail up to the micro-architecture level. Noxim provides flexibility to specify many different properties of a NoC such as the routing algorithm (deterministic or adaptive), arbitration policies, buffer size, etc. In Noxim a wormhole switching router is modeled. We have modified the Noxim simulator to implement the LBDR mechanism needed to evaluate our virtualization proposal.

In addition, architectural power estimation is extremely important in order to verify that power budgets are approximately met by the different components of a certain design, and evaluate the effect of the different proposals. For this reason, we have integrated the Orion 2.0 [31] power model inside the Noxim simulator in order to estimate the energy and area consumption of the simulated NoCs.

4.2. System integration

Much of the work developed for obtaining the full-system simulator is the inclusion of a new model of interconnection network inside of the Simics–GEMS system. To include the Noxim network, we have developed the *INetwork* interface. This interface also allows to include any network simulator in a simple way, and is highly independent of the Noxim simulator.

Ruby uses a queue-driven event model to simulate timing. Although many buffers are used in a strictly FIFO manner, the buffers are not only restricted to FIFO behavior. The simulation proceeds by invoking the *wakeup*() method for the next scheduled event on the event queue. The *INetwork* class produces events that are injected in the queue-driven event of the Ruby module, and these events implement the *wakeup*() method for our purpose.

In a global view, the transactions (load/store) are produced as consequence of the application execution in Simics. To satisfy a load or a store, each transaction is composed by several Ruby messages. When a transaction occurs, the driver sends a wait signal to Simics. The next step consists in simulating the messages over the CMP modeled through the Noxim network and the Ruby memory hierarchies. Finally, after that, the obtained latency (due to memory and network) must be returned to Simics. The process continues with other transaction and so on.

The *wakeup*() algorithm can be seen in Algorithm 1. For each input port, and for each *message* that needs to be simulated in the network, the *wakeup*() method is the responsible to extract the *message* (line 3), and transform the ruby *message* to a Noxim *packet* (line 4). Then, the *packet* is injected in the Noxim network (line 6), but previously the *packet id* is stored in the system (line 5) while the packet is simulated in the Noxim network.

Algorithm 1. [wakeup(noximNetwork) pseudocode.]

```
1: for all input ports do
2:     for all message in the port do
3:         message← get message from ruby
4:         packet← transform(message)
5:         stored packet id
6:         inject into noxim network (packet)
7:     end for
8: end for
9: put on noxim signal clock (1 cycle)
10:for all node of the CMP do
11:     for all received packet do
12:         packet← get packet from noxim
13:         id← get id (message)
14:         inject into ruby (id)
15:     end for
16:end for
17: ruby event queue ← noximNetwork event
```

The next step consists in executing one cycle the Noxim network and, subsequently, the *wakeup*() method performs the inverse process described above. That is, for each node of the simulated CMP, and for each received *packet* from Noxim, the *wakeup*() method extracts the *id* of the received *packet* from the corresponding node (lines 12 and 13), and informs to ruby of its reception in the simulated network. Finally, the noximNetwork event is injected in the global ruby queue.

Finally, we have extended the Noxim simulator with the Orion 2.0 model [31] in order to provide the power and area estimated from a given simulation. The result is a new NoC simulator that provides the typical evaluation measurements such as the throughput, latency, as well as the area and power-consumption. The integration consists in merging both simulators. On the one hand, the Noxim simulator must account for all the actions involving energy consumption, for instance, to transmit a flit over the links, to forward a flit in one router, to store a flit in the input/output ports, etc. On the other hand, the Orion simulator estimates the energy and the area consumption depending on the configurable parameters such as the technology, operating frequency, and operating voltage.

4.3. The CMP model

We model a homogeneous CMP with in-order cores. The CMP is structured in a number of nodes (tiles), each with a processing

Please cite this article in press as: F. Triviño et al., Virtualizing network-on-chip resources in chip-multiprocessors, Microprocess. Microsyst. (2010), doi:10.1016/j.micpro.2010.10.001

element, a router, a private L1 cache, a shared L2 cache, a memory directory bank, and a memory controller. The L2 cache is physically distributed but logically shared at the chip level. In addition, the memory directory bank is directly connected to the node. Coherence between L2 and L1 is kept using a directory-based MOESI protocol. We assume 3D-stacking [32] and so each node has a memory controller with a single DDR-2 memory channel. Finally, in order to avoid the interference among the L2 cache of nodes of different regions, we use a mapping distributed through OS-level page allocation [33] that forces the use of the L2 cache blocks that belong to the region where the application is mapped. Fig. 7 shows a general view of the CMP model. In Tables 1 and 2, we include the simulation parameters modeling the complete CMP architecture.

Regarding the on-chip network, we use a mesh topology which has all the links among nodes of the same size and width. Specifically, flit size is equal to the network link width, and we use a 4-byte flit size. Since cache lines are 64 bytes, we model 8-byte control request messages and 72-byte responses, so that requests are 2 flits long and responses 18 flits long. We use the LBDR mechanism for creating the region and domain definitions in the mesh, as it is shown in Fig. 4. We also use the Segment-Based Routing algorithm (SR [34]) in order to easily prevent cyclic dependencies in each virtual region.

4.4. Workload

As workload we have used the PARSEC v*2.1* benchmark suite [35]. This benchmark suite contains emerging applications and commercial solutions that cover a wide area of working sets and allows current CMP technologies to be studied more effectively [36]. The new version consists of nine applications and three kernels where each one has been parallelized and focused on emerging workloads. The workloads are diverse and were chosen from many different areas such as computer vision, media processing, computational finance, enterprise servers, and animation physics.

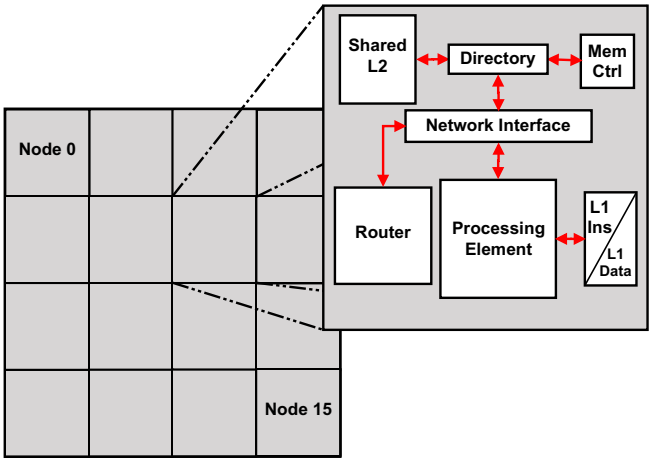


Fig. 7. The general scheme of a CMP with 16 nodes.

Table 1
Overview of the CMP configuration.

Parameter	Configuration(s)
OS	Solaris 10
CPU	16 × UltraSparc III processor
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
Coherence protocol	MOESI-CMP-Directory

Table 2
Overview of the on-chip network configuration.

Parameter	Configuration(s)
Topology	2D-mesh regular 4x4
Routing algorithm	LBDR with SR
Flit size	4 bytes
Buffer queue size	4 flits
Flow control	ABP flow control
Network frequency	The NoC works at the same frequency as the processors
Process technology	32 nm
Transistor type	LVT for high performance router
Operating voltage	1.2
Wire layer type	Intermediate
Wire layer length	100 μm

Each one of the workloads defines six input sets with different properties. The input sets *Test* and *Simdev* are intended for testing the basic functionality of the workload. *Simsmall*, *Simmedium* and *Simlarge* define different size for simulations. Finally, *Native* is the largest input set and corresponds with the real application execution.

We have evaluated the NoC virtualization proposals using all the applications from the PARSEC v2.1 benchmark, but due to the lack of space, we only show results for two sets of applications. Table 3 shows the application domain and the computational workload to run each application. In both sets, the Blacksholes application is running together with the Fluidanimate and Bodytrack applications for the first set, and with Swaptions and Streamcluster applications for the second set, all of them configured with *Simmedium* inputs. However, the results for other combinations obtained from the rest of applications and input sets are similar. The last row of each set of applications (Table 3) shows the synthetic traffic used to stress the network.

The stress traffic is only composed of artificial messages in the network but they do not represent any computational workload in the processing elements, not even cache usage. We use the stress traffic in order to show the tendency in the application's performance when the traffic interferences increase due to different network load. For this stress traffic, we have selected three different packet injection rates (PIR). This value represents a constant message generation rate at the nodes. For instance, a PIR value of 0.1 means that a flit will be injected in the interconnection

Table 3
Selected workloads.

Set	Application	Domains	Simmedium Input
One	Blackscholes	Financial analysis	16,384 options
	Fluidanimate	Animation	100,000 particles, five frames
	Bodytrack	Computer vision	2000 particles, four cameras, Five annealing layers
	Traffic stress	Traffic model	0.1, 0.2, 0.3 PIR
Two	Blackscholes	Financial analysis	16,384 options
	Swaptions	Financial analysis	32 swaptions, 10,000 simulations
	Streamcluster	Data mining	8192 input points, 64 point dimensions
	Traffic stress	Traffic model	0.1, 0.2, 0.3 PIR

Please cite this article in press as: F. Triviño et al., Virtualizing network-on-chip resources in chip-multiprocessors, Microprocess. Microsyst. (2010), doi:10.1016/j.micpro.2010.10.001

network each 10 cycles, assuming a peak bandwidth of 1 flit per cycle. The pair source–destination will be calculated randomly as well as the message size (from 8 to 72 bytes). For the low-medium rates (0.1 and 0.2) this traffic could represent one or more applications while for high rates (0.3) could represent an application that overloads the network.

This traffic also permits us to analyze the trend in the results when the network load increases. When we enable the virtualization mechanisms, the stress traffic is only generated in the nodes that form the region/domain. In the VR proposal the traffic is bounded to its region in order to eliminate the interferences among applications. On the other hand, in the VD proposal the messages of the stress traffic can cross the boundaries of its domain.

4.5. Scenarios

In order to evaluate our proposals, we have considered different scenarios. All of them assume the CMP model described in Section 4.3. In each scenario, three different PARSEC applications run simultaneously in the CMP, together with the stress workload. In order to avoid the effect of the scalability/parallelization degree, we have configured the three applications with four threads each one, as well as the stress traffic is mapped on four nodes. In this case, the scenarios only differ in the way they use the CMP resources. Note that our mechanism could manage different region sizes at same time without changing the hardware.

The first scenario we have considered is a baseline NoC with no mechanism to isolate the traffic of applications. In the baseline scenario (BL) each application uses the entire NoC and it will be allocated with four threads. The threads of each application will be distributed among the nodes through a random distribution. In this experiment, the traffic in the NoC generated by an application is affected by the traffic generated by the other applications and the stress traffic because all of them share and use the whole network. Fig. 8 shows an example of how the applications and the stress traffic are randomly mapped onto the whole CMP for both sets of workloads.

In the second scenario we enable the VR mechanism to create regions (VR). In this scenario, we assume the same CMP configuration than the BL scenario but, in this case, we have divided the NoC in four partitions. Fig. 9 shows different ways to divide a NoC in four regions. These and other different shapes can be obtained in the CMP as result of applying the VR mechanism in a more general dynamic scenario, when many applications enter and leave continuously the CMP system. In these scenarios each application is assigned to one region. Note that each region has the same number of resources but different shape. The applications are assigned with 4 threads, one per each node of the CMP, as in the BL scenario. In these schemes (Fig. 9a–d), traffic from one region is not allowed to traverse other regions. In these cases the stress traffic will be also isolated into the regions. The destinations of the messages can only be nodes inside the region.

Finally, we have considered the Virtual-domains scenarios (VD) for the same cases of the VR scenario. We called these scenarios as follow: A-VD, B-VD, C-VD, and D-VD. In these cases, the applications and the CMP are configured in the same way as in the VR scenarios. That is, for each scenario, the four applications are isolated in different domains that contain the same amount of resources. However, the messages belonging to one domain can cross the boundaries of its domain to reach their destinations. Fig. 10 shows how the VD mechanism is used in the B-VD case. Thus, the network load is distributed in a better way. Finally, as in the previous case, the stress traffic generates messages between nodes belonging to the same region. The stress traffic can only be nodes from the stress domain (domain number 4 for the set-1, and domain number 3 for the set-2). Note that in Fig. 10b each application has assigned the same nodes than in the B-VR case (Fig. 10a), although the messages generated by these applications can go around the dotted areas of Fig. 10b (minimal paths are assumed). Finally, although the other corresponding VD scenarios are not shown in the figure, we will take them into account in the evaluation results.

As a final consideration, the shapes of the regions for the VR and VD scenarios (Figs. 9 and 10) have been chosen as an example to show the possible interferences of different applications running together, except for the scenario A which presents no differences between VR and VD mechanisms. We have also performed tests with other region shapes but we have obtained similar results.

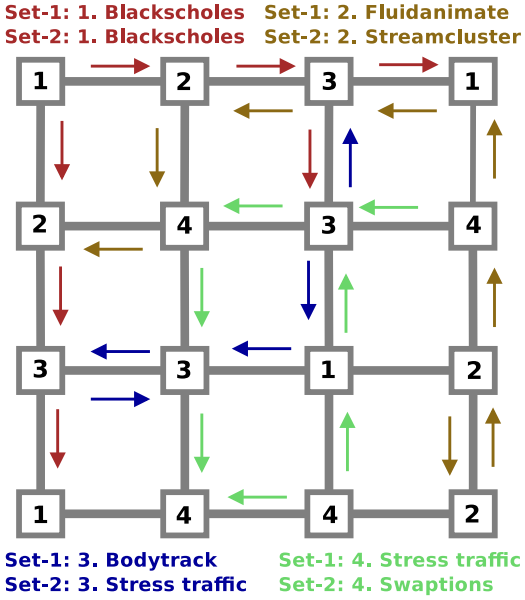


Fig. 8. Baseline scenario (**BL**).

5. Experimental results

In this section, we show the experimental results obtained in the evaluation process. We have considered static scenarios where three applications and the stress traffic are present at the same time in the system. In a more realistic scenario, the system would dynamically allocate the resources to the applications. When an application ends, the network resources it maintained would be freed, and so the system would reallocate them to new applications, maybe reallocating the still running applications in order to form regions with other shapes. In our static baseline scenario (threads randomly scattered through the nodes of the CMP) when the first application finishes, its resources would be used by the other applications still running in the same system, taking advantage of having more available resources. This would be unfair in order to compare the behaviors for the different scenarios. So, in order to avoid this effect, in all the experiments, we have collected statistics since the applications start until the end of the first application.

We have repeated three times the simulation of each scenario, varying the packet injection rate (PIR) of the stress traffic to evaluate the loss in performance due to the interaction of the messages of different applications at different load levels. This permits us to measure the application's performance when traffic load increases. Moreover, each value shown in the figures is the average of 30 different simulations, and the confidence interval has been set to 95%.

Please cite this article in press as: F. Triviño et al., Virtualizing network-on-chip resources in chip-multiprocessors, Microprocess. Microsyst. (2010), doi:10.1016/j.micpro.2010.10.001

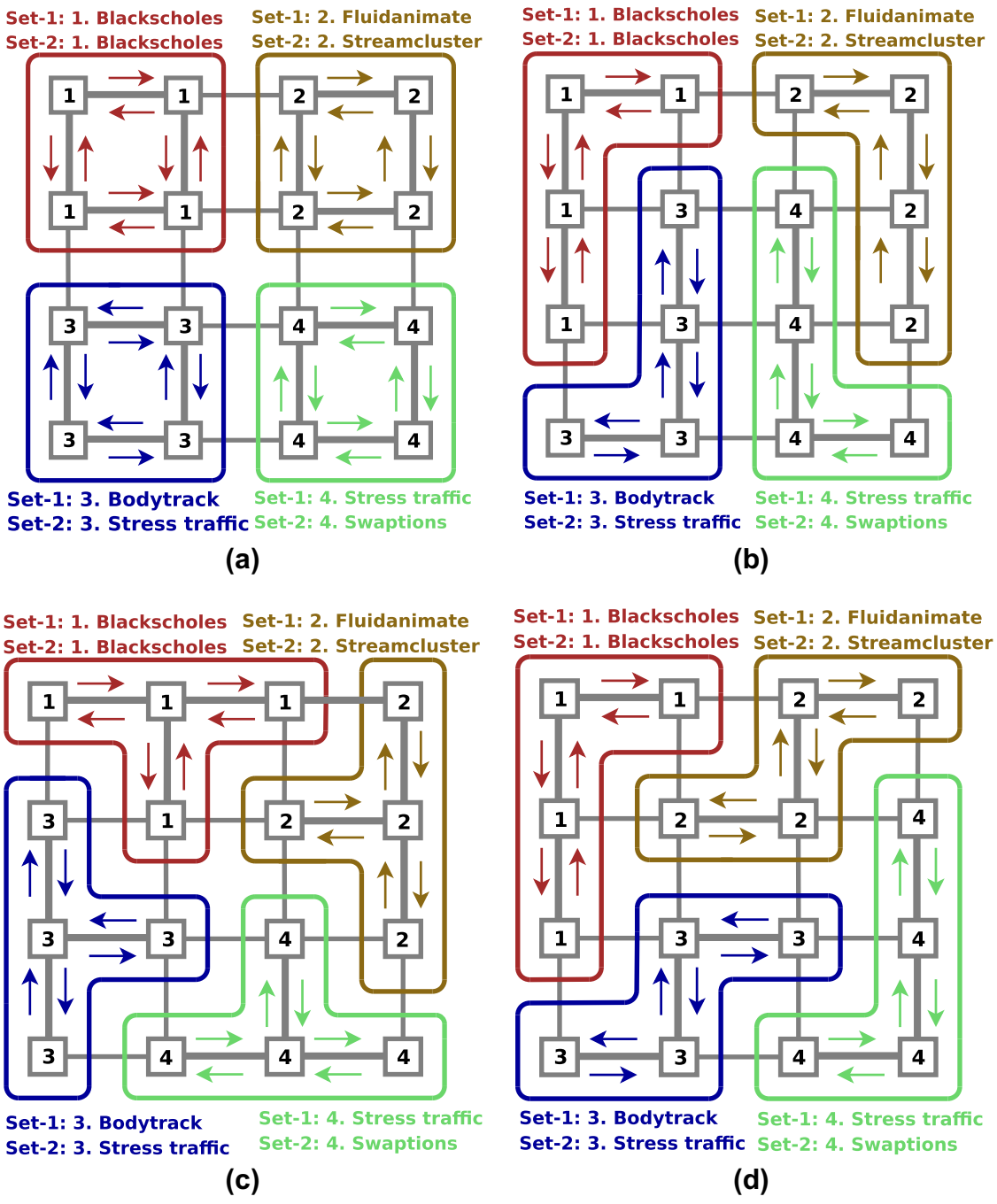


Fig. 9. (a) A-VR scenario, (b) B-VR scenario, (c) C-VR scenario, and (d) D-VR scenario.

On the other hand, in the BL scenario the application's threads are mapped in a random way. For this reason, we have repeated the simulations of this scenario 30 times with different mappings.

As already mentioned, we take statistics when the first application ends, and in our case this application is Blackscholes for both applications sets. As also mentioned before, we have repeated these tests with other applications, but we have obtained similar results.

5.1. Execution time

To begin with, it is interesting to analyze how the execution time evolves. Fig. 11 shows the normalized execution time of the different scenarios for both applications sets. The *x*-axis represents the obtained values for the BL scenario and for the virtualization scenarios from A-VR, A-VD to D-VR, D-VD. The cases A–D correspond with the shapes shown in the Fig. 9. The scenarios X-VR represent the virtual-regions mechanism applied to the shape X, and X-VD the Virtual-domains mechanism applied to the shape X. The figure represents the normalized execution time for three different PIR values (0.1, 0.2, and 0.3 PIR) of the stress traffic. Results are shown in normalized terms with respect to the execution time of the BL case for a 0.1 PIR value. That is, this value represent 100% and the rest of values are normalized considering this value.

In the BL case, all the applications must compete for the network resources as all of them are spread over the

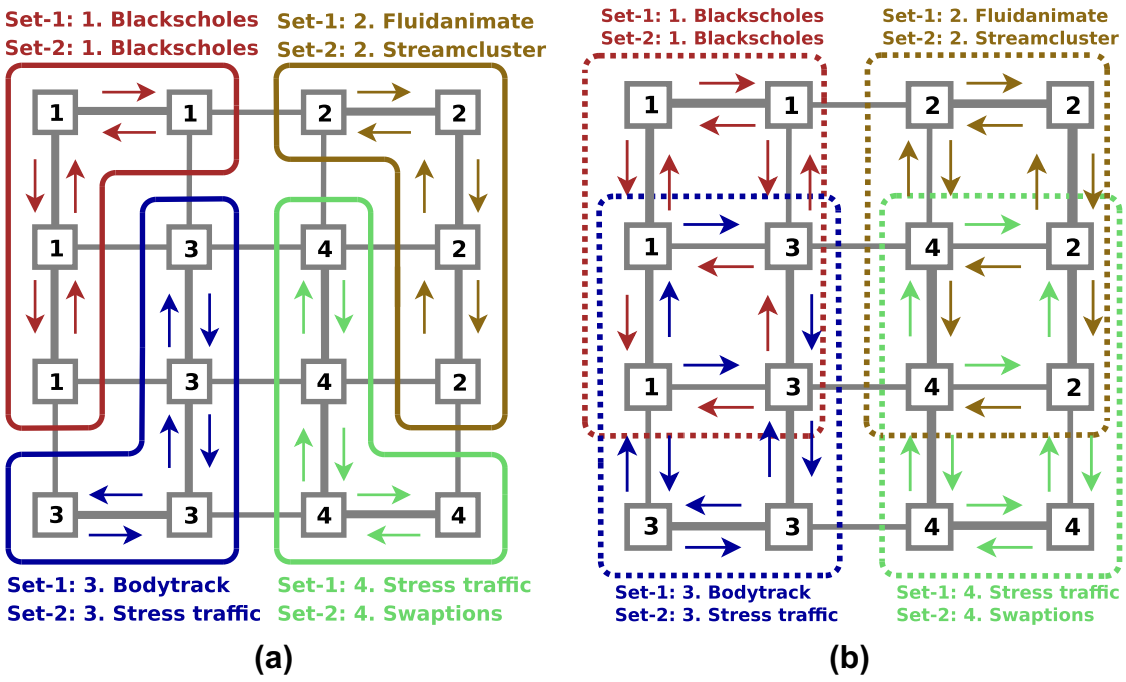


Fig. 10. The same shapes for the four regions in the CMP of the scenario B: (a) B-VR scenario and (b) B-VD scenario. Note that in (b) the messages belonging to one domain can cross the boundaries of its region in order to reach their destinations.

CMP,[3] thus sharing the network. Consequently the interferences affect all the applications, including Blackscholes. In that sense, we obtain a chaotic behavior as traffic interferences always occur among the applications. When multiple applications share the network, the communication cost due to the interferences among messages affects the performance of individual applications.

Note that as PIR value increases there are no variations on the execution time in the VR cases because all the applications and the stress traffic are isolated in different virtual-regions. Note also that there are no variations on the execution time between the A-VR and A-VD scenarios for any PIR value. This is because due to the shapes of the scenario A in these cases the regions/domains completely isolate the application and the stress traffic and so there are no opportunity to cross messages belonging to different domains.

Specifically, the execution time of the Blackscholes application is increased by 24% approximately from 0.1 to 0.3 PIR value as seen from Fig. 11a which corresponds with the set-1 of applications. Regarding the Fig. 11b (set-2), this increment is 18%. As expected, when we use the virtualization approaches the traffic has lower interferences and so the performance is much better and stable. For instance, if we take the best virtualization case (A-VR and A-VD) as a reference, the figure shows a reduction of 25% in the execution time compared to the BL case, when the PIR value is set to 0.1. Moreover, as the PIR value is increased the difference between both cases (BL vs. VR/VD) increases.

In the other cases (B–D), the execution time differences among virtualization scenarios depend on the region/domain shapes and the stress load. For instance, as PIR value increases there are no variations on the execution time in the VR cases because all the applications and the stress traffic are isolated in different regions. There is a little difference between the VR and VD cases shown in Fig. 11a (set-1). In this case, due to the assignment of the applications to the domains (see Fig. 9), the final execution time of the Blackshcoles application is only affected by the traffic of the Bodytrack application (its packets can cross the boundaries of its region) but not by the stress traffic (in the set-1). For this reason, the execution time of the Blackscholes application for the VD case is higher than that obtained in the VR case, and this value does not vary for different PIR values. This is not true for the set-2 of applications because, in this case, the stress traffic is mapped in the domain 3 and the execution time of the VD cases varies for different PIR values.

On the other hand, due to the particular characteristics of the synthetic traffic, one could think if the stress traffic is altering the obtained results and they would be different just with real applications. For this reason, we have performed a simulation only with real applications in order to show the trend from a simulation without synthetic traffic.

Fig. 12 shows again the normalized execution time for the applications set-2 when the stress traffic is configured as 0.1 and 0.2 PIR values, again in normalized terms compared with the obtained value for the BL case when it uses a 0.1 PIR value. Moreover, in this figure, we have added the results of one simulation whose workload is composed of four PARSEC applications instead of three applications and the stress traffic configured at 0.3 PIR like in Fig. 10. That is, we have replaced the stress traffic by the Fluidanimate application. Therefore, the selected applications are Blackscholes, Swaptions, Streamcluster, and Fluidanimate. We have referred these results in the figure as “Four Applications”.

The results for this experiment show the similarity with the results obtained in the stress traffic cases. In the new scenario, for the BL case, we can see an increment of 4% compared with the 0.1 PIR case, and a decrement of 5% compared with the 0.2 PIR case. For the virtualization cases, the trend in the results is also maintained. As can be seen, when running four applications simultaneously, the results are quite similar to configure the stress traffic between 0.1 and 0.2 PIR values. As a final consideration, we have tested this experiment with other sets of applications and the results are quite similar.

[3] The four threads of each application are randomly distributed in the CMP.

Please cite this article in press as: F. Triviño et al., Virtualizing network-on-chip resources in chip-multiprocessors, Microprocess. Microsyst. (2010), doi:10.1016/j.micpro.2010.10.001

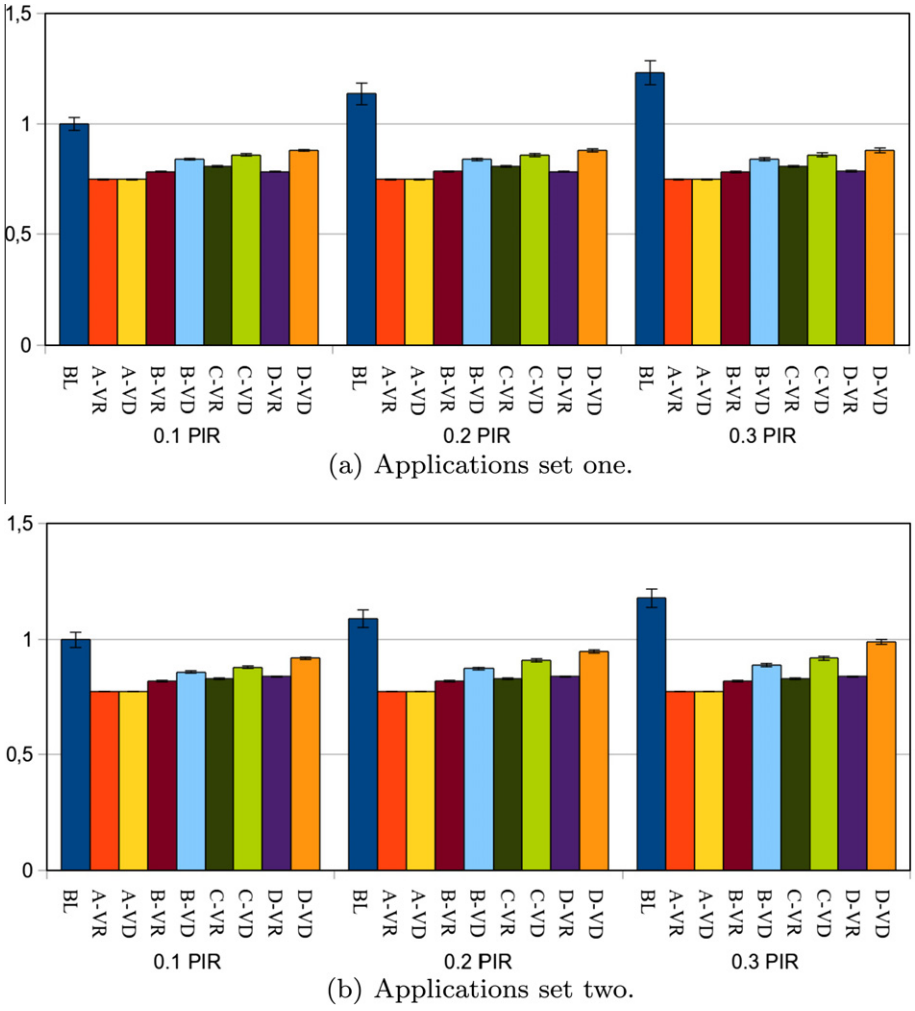


Fig. 11. Normalized execution time.

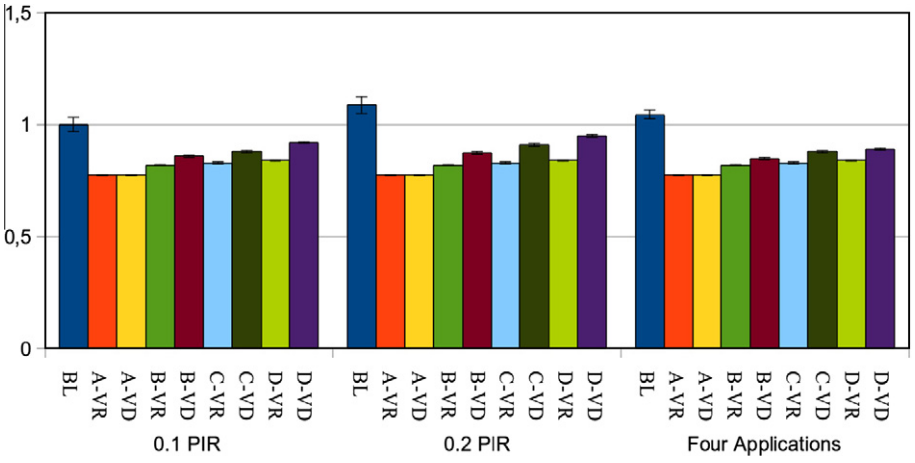


Fig. 12. Normalized execution time.

Therefore, these results strengthen our conclusions showing that real applications disturb neighboring applications as if we were using stress traffic of 0.1 or 0.2 PIR values, depending on the application. So, our methodology of using stress traffic is perfectly correct and permits us to check the trend when traffic load increases. In the following sections only this methodology will be used: three applications and the stress traffic because no differences are obtained when four real applications are used instead of three real applications and the stress traffic. Moreover, the use of different PIR values of stress traffic permits us to study the trend of the results when network load varies.

Please cite this article in press as: F. Triviño et al., Virtualizing network-on-chip resources in chip-multiprocessors, Microprocess. Microsyst. (2010), doi:10.1016/j.micpro.2010.10.001

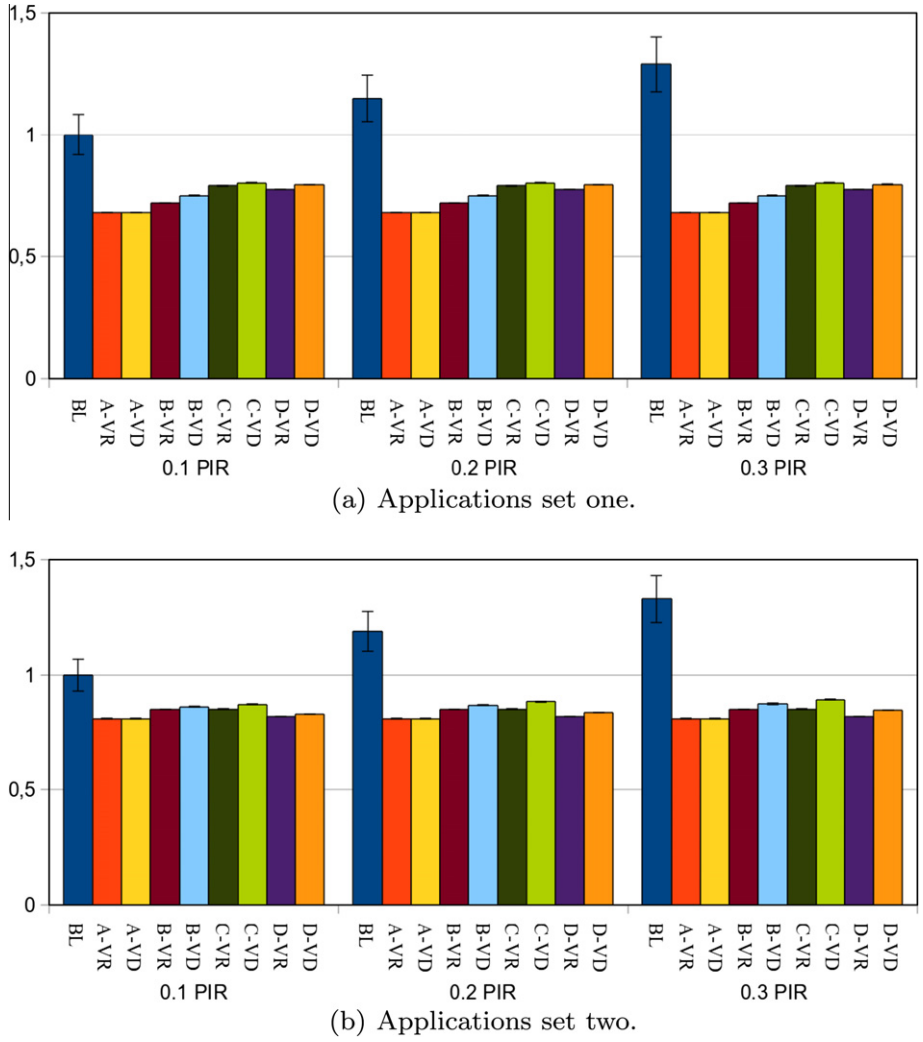


Fig. 13. Normalized network latency.

5.2. Network latency

Although the execution time is the most important metric when we use real applications, in the interconnection networks domain other important metrics as the network latency and the network throughput are used. Fig. 13 shows the network latency for both applications sets, again in normalized terms compared with the obtained value for the BL case when it takes a 0.1 PIR value. In the set-1 case (Fig. 13a), as the PIR value increases, the latency obtained for the BL case increases (29% from 0.1 to 0.3 PIR), but in the VR and VD cases the latency keeps constant. As expected, in the set-2 case (Fig. 13b) the latency is also increased for the BL scenario (33% from 0.1 to 0.3 PIR) while the latency for the VD scenario does not keep constant (it increases) due to the fact that the stress traffic affects the Blackscholes traffic because both traffics are sharing network resources (only in the VD scenarios). As a consequence, the interferences affect the obtained latency of the blackscholes application.

The VR/VD cases get lower latency values compared with the BL scenario (a reduction of 32% in the set-1 and of 19% for the set-2 for the A-VR cases), because they divide the network into different regions/domains and the average message distance is significantly reduced. Thus, the traffic of each application has a very low latency, which is also one of the reasons of obtaining a lower execution time. When source and destination nodes are not placed adjacent to each other on the network, a packet needs to travel several intermediate nodes until reaching the destination. When the number of hops increases the probability of interference with other packets also increases, which turns in an increment of the latency. Finally, note that the latency does not vary when we increase the PIR value in the set-1, but it does for the set-2 again because in that case the stress traffic affects the Blackscholes execution.

Finally, as can be observed in the figure, the trend for the network latency is the same that the trend showed for the execution time.

5.3. Network throughput

Fig. 14 shows the network throughput, again in normalized terms compared with the obtained value of the BL case when it takes a 0.1 PIR value. We measure the throughput in flits per cycle. In the BL scenario we observe that the throughput slightly decreases. When PIR value increases we must consider two facts. The first one is that the execution time increases considerably (as we can see in the Fig. 11), therefore, this fact should reduce the network throughput in a significant amount. However, when the PIR value increases both the interferences among messages and the cache miss rate also increase (as it was shown in Fig. 1), and

Please cite this article in press as: F. Triviño et al., Virtualizing network-on-chip resources in chip-multiprocessors, Microprocess. Microsyst. (2010), doi:10.1016/j.micpro.2010.10.001

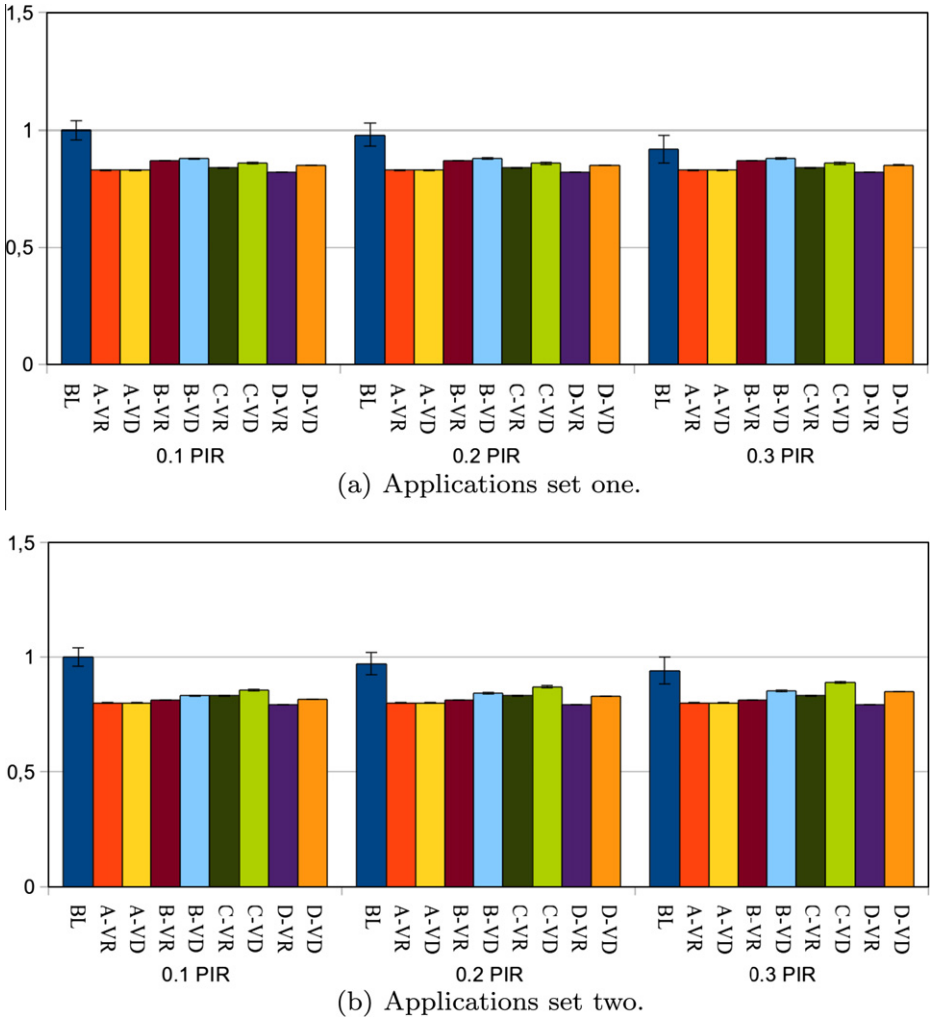


Fig. 14. Normalized network throughput.

as a consequence the total amount of flits to transmit increases too. Therefore, the important increase of the execution time is compensated by the increase in the amount of flits being transmitted due to the higher miss rate. As can be observed, as the PIR value increases the throughput obtained for the BL case slightly decreases, being this decrement of 8% for the case of 0.3 PIR value for the applications set-1 and of 6% for the applications set-2.

On the other hand, the network throughput is lower for the virtualization approaches than in the BL case: the A-VR/A-VD cases compared to the BL case show a reduction of 18%, approximately. This is due to the different amount of transmitted flits in both cases (BL and A-VR/A-VD). Regarding the B-VD, C-VD, and D-VD cases compared to the corresponding VR cases, the interferences in the VD cases are increased because part of the network is shared between the Blackscholes and the Bodytrack workloads in the set-1 of applications (Fig. 14a). As a consequence the number of misses increases and this generates more traffic for the same applications and for this reason the throughput is slightly higher than in the VR case. This behavior also occurs for the set-2 of applications (Fig. 14b) but, in this case, the network is shared between the Blackscholes application and the stress traffic.

5.4. Energy consumption

Fig. 15 shows the network energy consumption obtained from the Orion power simulator taking into account the application traffics as well as the stress traffic. As can be seen, the BL scenario has a higher energy consumption. The reason can be found in the fact that the BL scenario has a higher utilization of the network resources and that the simulation takes more time to finish the Blackscholes execution when we increase the PIR value. Therefore the Blackscholes application is running in the CMP more time and thereby the energy consumption is increased. Note also that in the VR and VD scenarios the unused links are switched off. For the same shapes of the regions/domains the number of unused links is lower in the VD scenarios than in the VR scenarios due to the fact that in the VD cases the messages can cross the boundaries of their region.

As mentioned before, the VR/VD cases have a lower energy consumption compared to the BL case. Specifically, the energy consumption is decreased by 13% approximately as seen from Fig. 15a which corresponds with the set-1 of applications and by 10% for the set-2 of applications (Fig. 15b) in comparison to the best virtualization case. This was expected as the VR/VD scenarios finish the Blackscholes application earlier. Between the VR and VD scenarios the differences are also based on the differences on the execution time and the different way of using the network resources. Note that for all the VD scenarios, the energy consumption is higher than for the VR cases because they involve more traffic. The increment of energy consumption is related to the geometries of the regions/domains and also to the unused links. In the particular case of an injection rate of 0.3 PIR and the set-1 of applications,

Please cite this article in press as: F. Triviño et al., Virtualizing network-on-chip resources in chip-multiprocessors, Microprocess. Microsyst. (2010), doi:10.1016/j.micpro.2010.10.001

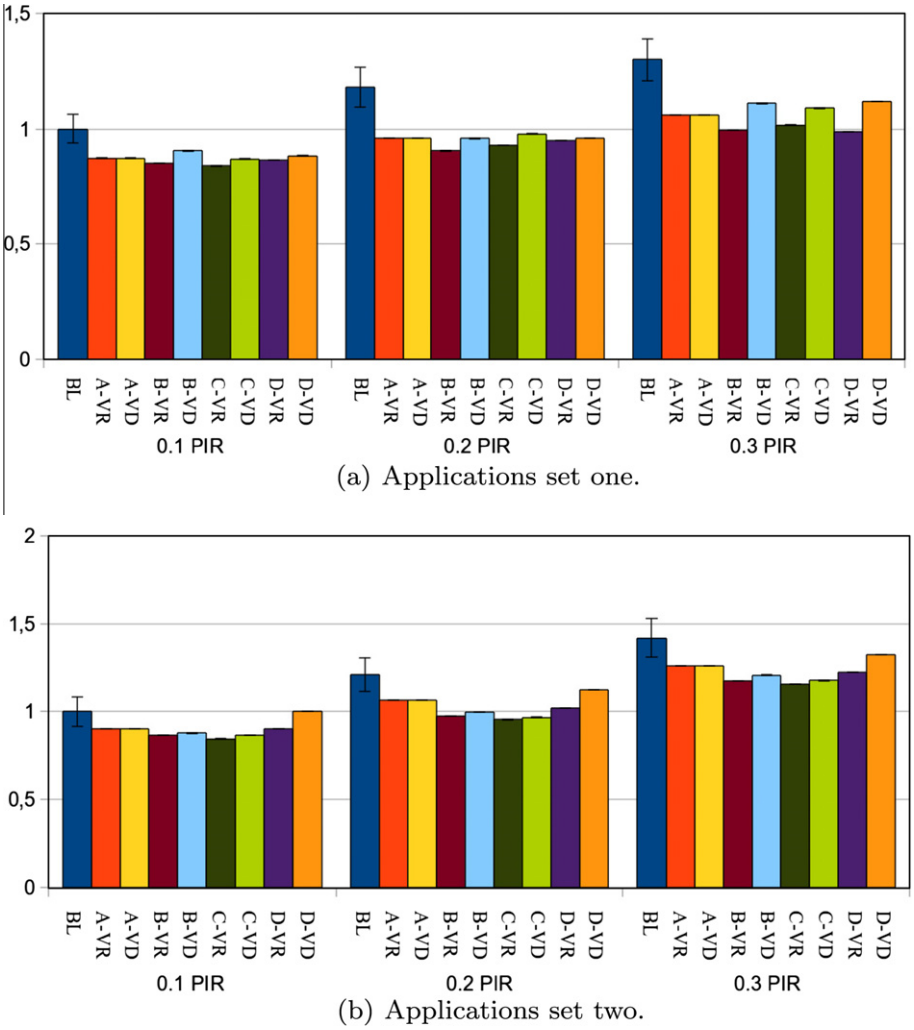


Fig. 15. Normalized energy consumption.

the energy consumption of the B-VR case has increased 12% over the B-VD case, and 9% for the C-VR compared to the C-VD case, approximately.

5.5. Link utilization

Finally, we are also interested in studying how the network traffic is distributed among the links that compose the NoC. Since we have modeled a 2D 4 × 4 mesh, the number of links is 40. The 24 global links are used for connecting the routers, and the remainder (16 internal links) are used to interconnect the cores with the routers. We show the utilization percentage of each link for each scenario from the start of the simulation until the simulation stops. Thus, this information shows the percentage of cycles that the links have been used. Since we have repeated the simulation of each scenario 30 times (where we have only changed the seed), we show the mean of each link usage. We use frequency histograms in order to summarize these results. These histograms contain the link percentage of usage taking into account all the traffic generated by the applications and the stress traffic.

Fig. 16 shows the link usage for all the different schemes with a 0.2 PIR value[4] for the stress traffic, and for the set-1 of applications. Finally, the *x*-axis represents the utilization percentage per ranges of 10% from 0% to 100%, while the *y*-axis represents the number of links that have a determined utilization shown in the *x*-axis.

As can be seen, the link utilization is higher for the BL configuration. There are no links with a frequency between 0% and 10% and most of the links are in the range of 30–40% of utilization. It seems logical since all the links in the CMP can be used in this case and paths can also be longer, so increasing the utilization. In contrast, some links are unused in the virtualization approaches, and so the average utilization decreases. The VR scenarios present the lowest link utilization level because a large number of links are between 0% and 10% of utilization. Some of them correspond with links that are not used because of the virtualization process, and thereby have a null utilization. For the VD scenarios, as a packet can cross the boundaries of its domain, some of these links are used and the utilization slightly increases. However, the differences between the virtualization models (VR and VD) are minimal because these cases are very restricted due to the shape of the regions. If the CMP size would be larger and there would be a greater number of regions, this difference would be higher.

6. Conclusions and future work

This paper aims to improve the performance of the applications that are running simultaneously in a CMP. Applications share CMP resources and performance of individual application can be

[4] A 0.2 PIR value represents a middle load for the links that the stress traffic uses.

Please cite this article in press as: F. Triviño et al., Virtualizing network-on-chip resources in chip-multiprocessors, Microprocess. Microsyst. (2010), doi:10.1016/j.micpro.2010.10.001

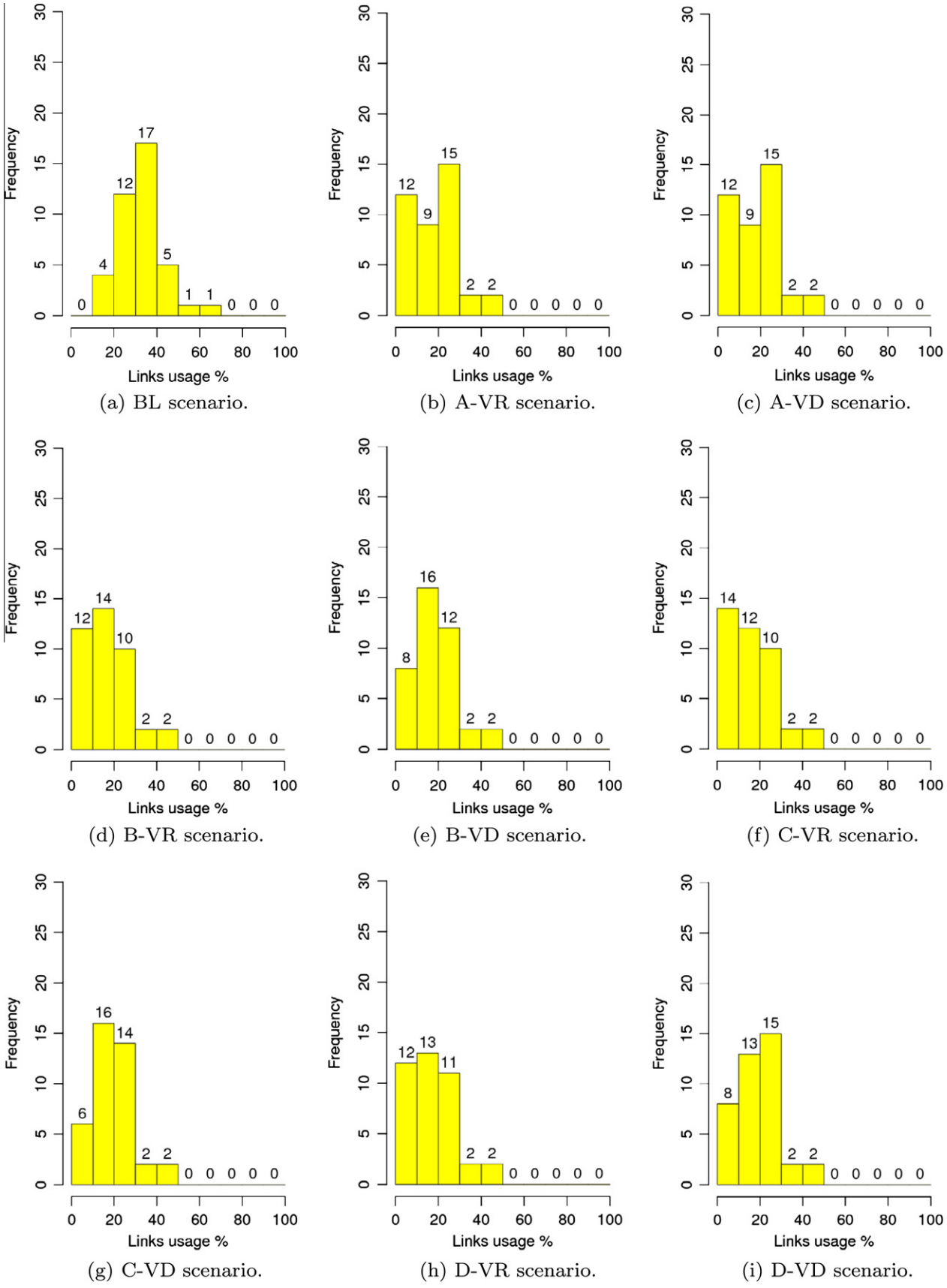


Fig. 16. Utilization of the links under 0.2 PIR stress traffic for the set-1 of applications. There are 40 links, 16 internal links connecting each processing element with the router and 24 links to communicate among routers.

Please cite this article in press as: F. Triviño et al., Virtualizing network-on-chip resources in chip-multiprocessors, Microprocess. Microsyst. (2010), doi:10.1016/j.micpro.2010.10.001

seriously affected. If we put our attention only in the interconnection network, we can see that traffic interferences affect the network performance and as a consequence the CMP performance. To address this issue, the network needs a mechanism to isolate the traffic in order to reduce or even eliminate the communication cost due to the interferences among messages belonging to different applications.

First of all, we have observed that for the baseline scenario (all the applications sharing the network of the CMP) the execution time and the network latency increase when the traffic load increases. Regarding the network throughput, for this baseline scenario it decreases when the traffic load increases. In contrast, for both virtualization approaches the execution time, the network latency and the network throughput keeps almost constant when the network traffic load increases.

In particular, we have shown that the best of our two virtualization approaches reduces by 25% the execution time of the Blackscholes application and 32% the network latency when compared to a NoC baseline configuration. Similar values are obtained for other applications that we have tested. This fact is due to two reasons: the elimination of the interferences among messages belonging to different applications and the lower average message distance in the virtualized approaches.

Regarding the network throughput, we obtain lower values for the virtualization approaches than for the baseline scenario (the VR/VD cases compared to BL show a reduction of 18%, approximately). The main reason is the difference between the execution time and the total amount of transmitted flits in both cases (VR/VD) compared to the baseline case. Although the execution time is reduced considerably, the total amount of transmitted flits is also reduced due to the elimination of the interferences among applications.

Again, the network energy consumption is reduced considerably in the virtualization approaches. Due to the fact that the baseline scenario has a higher utilization of the network resources and the execution time is higher than in the virtualization approaches, the virtualization cases get lower values of energy consumption.

Regarding the link utilization, the percentage of unused links due to the virtualization increases as more regions or domains are created. Instead of seeing that as a negative point, we think that can be seen as something positive if the rest of the links maintain far of saturation as it is this case. In this sense, the virtualization approaches offer an important advantage because the interconnect components that are unused can be switched off in order to reduce the power-consumption.

There are still open questions. In this work, we have performed static experiments. Future work is needed to evaluate strategies for assigning the total amount of resources in a dynamic way that simulates a real behavior. In a real scenario, the applications enter and leave the system, therefore the virtualization mechanism needs a resources manager which could form the regions/domains in a dinamic way and reconfigure them. We believe that, in order to meet the application requirements the regions must be dynamic and may change the total amount of their resources in real time.

Acknowledgments

This work was supported by the Spanish MEC and MICINN, as well as European Comission FEDER funds, under Grants CSD2006-00046 and TIN2009-14475-C04. It was also partly supported by Junta de Comunidades de Castilla-La Mancha under Grants PCC08-0078-9856, POII10-0289-3724, and by the project *NaNoC* (project label 248972) which is funded by the European Commission within the Research Programme FP7.

References

[1] V. Agarwal, M. Hrishikesh, S. Keckler, D. Burger, Clock rate versus IPC: the end of the road for conventional microarchitectures, in: ISCA, ACM, New York, NY, USA, 2000, pp. 248–259.
[2] Tilera Tile-Gx Product Brief. 2010. URL <http://www.tilera.com/pdf/PB025_TILE-Gx_Processor_A_v3.pdf>.
[3] Krewell K. Best Servers of 2004: Where Multicore is Norm. Microprocessor Report. 2005. URL <www.mpronline.com>.
[4] J. Howard, et al. A 48-Core IA-32 Message-Passing Processor with DVFS in 45nm CMOS, in: International Solid-State Circuits Conference, 2010.
[5] S.R. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, et al., An 80-tile sub-100-W TeraFLOPS processor in 65-nm CMOS, IEEE Journal of Solid-State Circuits 43 (1) (2008) 29–41.
[6] S. Bell, B. Edwards, J. Amann, R. Conlin, K. Joyce, V. Leung, et al. TILE64 processor: a 64-Core SoC with mesh interconnect, in: ISSCC. Digest of Technical Papers, IEEE International, 2008, pp. 88–598.
[7] P. Wielage, K. Goossens, Networks on Silicon: Blessing or Nightmare?, in: DSD, IEEE Computer Society, Washington, DC, USA, 2002, p 196.
[8] W.J. Dally, B. Towles, Route Packets, Not Wires: On-Chip Inteconnection Networks, in: DAC, ACM, New York, NY, USA, 2001, pp. 684–689.
[9] M. Sgroi, M. Sheets, A. Mihal, K. Keutzer, S. Malik, J. Rabaey, et al., Addressing the System-on-a-Chip Interconnect Woes Through Communication-Based Design, in: DAC, ACM, New York, NY, USA, 2001, pp. 667–672.
[10] K. Goossens, P. Wielage, A. Peeters, J. Van Meerbergen, Networks on Silicon: Combining Best-Effort and Guaranteed Services, in: DATE, IEEE Computer Society, Washington, DC, USA, 2002, p. 423.
[11] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, et al., On-chip interconnection architecture of the tile processor, IEEE Micro 27 (5) (2007) 15–31.
[12] G.D. Micheli, L. Benini, Networks on Chips: Technology and Tools (Systems on Silicon), Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2006.
[13] D. Gupta, L. Cherkasova, R. Gardner, A. Vahdat, Enforcing Performance Isolation Across Virtual Machines in Xen, in: Middleware, Springer-Verlag New York, Inc., New York, NY, USA, 2006, pp. 342–362.
[14] V. Subramani, R. Kettimuthu, S. Srinivasan, J. anette Johnston, P. Sadayappan, Selective buddy allocation for scheduling parallel jobs on clusters, Cluster 0 (2002) 107.
[15] P.J. Chuang, N.F. Tzeng. An efficient submesh allocation strategy for mesh computer systems, in: ICDCS, 1991, pp. 256–263.
[16] Li K, Cheng KH. A two dimensional buddy system for dynamic resource allocation in a partitionable mesh connected system, in: CSC. ACM, New York, NY, USA, 1990, pp. 22–27.
[17] Y. Zhu, Efficient processor allocation strategies for mesh-connected parallel computers, Journal of Parallel and Distributed Computing 16 (4) (1992) 328–337.
[18] M.R. Marty, M.D. Hill. Virtual hierarchies to support server consolidation, in: ISCA, ACM, New York, NY, USA, 2007, pp. 46–56.
[19] F. Guo, H. Kannan, L. Zhao, R. Illikkal, R. Iyer, D. Newell, et al., From chaos to QoS: case studies in CMP resource management, SIGARCH Computer Architecture News 35 (1) (2007) 21–30.
[20] J. Flich, J. Duato, T. Sødring, Å.G. Solheim, T. Skeie, O. Lysne, et al. On the potential of NoC virtualization for multicore chips, in: MuCoCoS. IEEE, 2008, pp. 801–807.
[21] K.K. Paliwal, M. Gaur, V. Laxmi, V. Janyani, Performance analysis of guaranteed throughput and best effort traffic in network-on-chip under different traffic scenario, ICFN 0 (2009) 74–78.
[22] J. Liang, S. Swaminathan, R. Tessier, aSOC: a scalable, single-chip communications architecture, PACT 0 (2000) 37.
[23] F. Karim, A. Nguyen, S. Dey, An interconnect architecture for networking systems on chips, IEEE Micro 22 (5) (2002) 36–45.
[24] K. Goossens, J. Dielissen, A. Radulescu, Ethereal network on chip: concepts, architectures, and implementations, IEEE Design and Test of Computers 22 (5) (2005) 414–421.
[25] J. Flich, J. Duato, Logic-Based Distributed Routing for NoCs, IEEE Computer Architecture Letters 7 (1) (2008) 13–16.
[26] J. Flich, S. Rodrigo, J. Duato, An efficient implementation of distributed routing algorithms for NoCs, NOCS (2008) 87–96.
[27] M. Schroeder, A. Birrell, M. Burrows, H. Murray, R. Needham, T. Rodeheffer, et al., Autonet: a high-speed, self-configuring local area network using point-to-point links, J-SAC 9 (8) (1991) 1318–1335.
[28] P.S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hållberg, J. Högberg, et al., Simics: a full system simulation platform, Computer 35 (2) (2002) 50–58.
[29] M.M.K. Martin, D.J. Sorin, B.M. Beckmann, M.R. Marty, M. Xu, A.R. Alameldeen, et al., Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset, SIGARCH Computer Architecture News 33 (4) (2005) 92–99.
[30] Noxim Simulator, 2009. URL <http://noxim.sourceforge.net/>.
[31] A.B. Kahng, B. Li, L.S. Peh, K. Samadi. ORION 2.0: a fast and accurate NoC power and area model for early-stage design space exploration, in: DATE, 2009, pp. 423–428.
[32] B. Black, M. Annavaram, N. Brekelbaum, J. DeVale, L. Jiang, G.H. Loh, et al., Die Stacking (3D) Microarchitecture, in: MICRO, IEEE Computer Society, Washington, DC, USA, 2006, pp. 469–479.

Please cite this article in press as: F. Triviño et al., Virtualizing network-on-chip resources in chip-multiprocessors, Microprocess. Microsyst. (2010), doi:10.1016/j.micpro.2010.10.001

[33] S. Cho, L. Jin. Managing distributed, shared L2 caches through OS-level page allocation, in: MICRO, IEEE Computer Society, Washington, DC, USA, 2006, pp. 455–468.
[34] A. Mejia, J. Flich, J. Duato. On the potentials of segment-based routing for NoCs, in: ICPP, IEEE Computer Society, Washington, DC, USA, 2008, pp. 594–603.
[35] C. Bienia, K. Li. PARSEC 2.0: A new benchmark suite for chip-multiprocessors, in: MoBS, 2009.
[36] C. Bienia, S. Kumar, K. Li. PARSEC vs. SPLASH-2: a quantitative comparison of two multithreaded benchmark suites on chip-multiprocessors, in: IISWC, 2008.

Francisco Triviño received the MS degree in computer science from the University of Castilla-La Mancha, Spain, in 2008 and is currently working toward the PhD degree. He is currently a research assistant in the Research Group in High Performance Networks and Architectures, University of Castilla-La Mancha. His research interests include network-on-chip and quality of service.

José L. Sánchez received the PhD degree from the Technical University of Valencia, Spain, in 1998. Since November 1986 he is a member of the Computer Systems Department (formerly Computer Science Department) at the University of Castilla-La Mancha. He is currently an associate professor of computer architecture and technology. His research interests include multicomputer systems, quality of service in high-speed networks, interconnection networks, networks-on-chip, parallel algorithms and simulation.

Francisco J. Alfaro received the MS degree in computer science from the University of Murcia in 1995 and the PhD degree from the University of Castilla-La Mancha in 2003. He is currently an assistant professor of computer architecture and technology in the Computer Systems Department at the Castilla-La Mancha University. His research interests include high-performance local area networks, QoS, design of high-performance routers, and design of on-chip interconnection networks for multi-core systems.

José Flich received his MS and PhD degrees in Computer Science from the Technical University of Valencia (Universidad Politécnica de Valencia), Spain, in 1994 and 2001, respectively. He joined the Department of Computer Engineering (DISCA), Universidad Politécnica de Valencia in 1998 where he is currently an Associate Professor of Computer Architecture and Technology. He has served as Program Committee member in different conferences, including ICPP, IPDPS, HiPC, CAC, ICPADS, and ISCC. He is currently co-chair of CAC and INA-OCMC workshops and vice-chair (high-performance networks track) of EuroPar conference. His research interests are related to high-performance interconnection networks for multiprocessor systems, cluster of workstations, and networks-on-chip.

Please cite this article in press as: F. Triviño et al., Virtualizing network-on-chip resources in chip-multiprocessors, Microprocess. Microsyst. (2010), doi:10.1016/j.micpro.2010.10.001