Scalable Computing: Practice and Experience
Volume 9, Number 3, pp. 165–177. http://www.scpe.org
ISSN 1895-1767

ON THE POTENTIAL OF NOC VIRTUALIZATION FOR MULTICORE CHIPS*

J. FLICH, S. RODRIGO, J. DUATO† T. SØDRING‡ Å. G. SOLHEIM, T. SKEIE, AND O. LYSNE§

Abstract. As the end of Moores-law is on the horizon, power becomes a limiting factor to continuous increases in performance gains for single-core processors. Processor engineers have shifted to the multicore paradigm and many-core processors are a reality. Within the context of these multicore chips, three key metrics point themselves out as being of major importance, *performance*, *fault-tolerance* (including yield), and *power consumption*. A solution that optimizes all three of these metrics is challenging. As the number of cores increases the importance of the interconnection network-on-chip (NoC) grows as well, and chip designers should aim to optimize these three key metrics in the NoC context as well.

In this paper we identify and discuss the main properties that a NoC must exhibit in order to enable such optimizations. In particular, we propose the use of virtualization techniques at the NoC level. As a major finding, we identify the implementation of unicast and broadcast routing algorithms to become a key design parameter in order to achieve an effective virtualization of the chip.

The intention behind this paper is for it to serve as a position paper on the topic of virtualization for NoC and the challenges that should be met at the routing layer in order to optimize performance, fault-tolerance and power consumption in multicore chips.

1. Introduction. Multicore chips (processors) are becoming mainstream components when designing high performance processors. As power is increasingly becoming a limiting factor with regards to performance for single-core solutions, designers are shifting to the multicore paradigm where several processor cores are integrated into one chip. Although the number of cores in current processing devices is small (i. e. two to eight cores per chip), the trend is expected to change. As an example, the Polaris chip for TeraFLOP computing from Intel has recently been announced with 80 cores [1]. This chip is a research demonstrator but serves to highlight many of the challenges facing NoC in the multicore era.

As the main processor manufacturers further integrate multicore into their product lines, an increasingly large number of cores is expected to be added to chips. In these chips, a requirement for a high-performance on-chip interconnect (NoC) emerges, allowing efficient communication between cores and from cores to cache blocks and/or memory controllers. Current chip implementations use simple network topologies such as buses or rings [2]. However, as the number of cores increases such networks effectively become bottlenecks within the system, and become impractical from a scalability point of view. For chips with a larger number of cores the 2D mesh topology is usually preferred due to its layout on a planar surface in the chip. This is also the case of the TeraScale chip. In this paper we restrict our view to 2D mesh and 2D torus topologies.

A lot of research has been undertaken on high performance networks over the recent decades and for some NoCs the mechanisms, techniques, and solutions proposed for off-chip networks can be applied directly. However, on-chip interconnects face physical constraints at the nanometer scale that are not encountered (or are not limiting solutions) in the off-chip domain. One example relates to the use of routing tables at the switches which is common for off-chip networks (e.g. InfiniBand [3]). For some NoCs, however, the memory requirements to implement routing tables in switches may not be acceptable.

From our point of view, the main constraints relating to NoCs (and that are not a primary concern in off-chip networks) are: power consumption, area requirements (floor-space), and ultra-low latencies. We note that these concerns suggest that simple solutions for NoCs should be implemented.

A multicore designer must deal with three key metrics when designing a multicore chip (see Figure 1.1): *performance*, *fault-tolerance*, and *power consumption/area*. Achieving a solution that optimizes these metrics is challenging. If we take fault-tolerance as an example, a solution for fault-tolerance may have a negative impact on performance resulting in increased latency. Such a solution may be acceptable and perhaps even desirable for an off-chip network but for a NoC it may be prohibitive. Another example is that for some NoCs, an efficient implementation of a routing algorithm that delivers high throughput may consume too many resources.

*This work was supported by CONSOLIDER-INGENIO 2010 under Grant CSD2006-00046, by CICYT under Grant TIN2006-15516-C04-01, by Junta de Comunidades de Castilla-La Mancha under Grant PCC08-0078, and by the HiPEAC network of excellence.

†Parallel Architectures Group, Technical University of Valencia, Spain.

‡Networks and Distributed Systems Department, Simula Research Laboratory, and Department of Journalism, Library and Information Science, University College Oslo, Norway

§Networks and Distributed Systems Department, Simula Research Laboratory, and Department of Informatics, University of Oslo, Norway

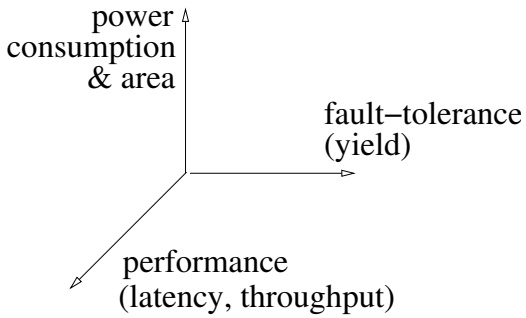


FIG. 1.1. *Design space for a NoC.*

The *Performance* of a network is traditionally measured in terms of packet latencies and network throughput. Currently, within the context of NoC, network throughput is not a major concern as wires can be widened to implement network links where needed. However, ultra-low latencies (less than a few nanoseconds) are typically required and it is usual that a switch forwards a flit within a network cycle. The implication of this is that every stage within the critical path of a packet must be carefully optimized. This is one of the reasons that, in some NoCs, logic-based routing (e.g. the predominant Dimension-Order-Routing, DOR) is the preferable solution as it can lead to reductions in latency as well as power and area requirements.

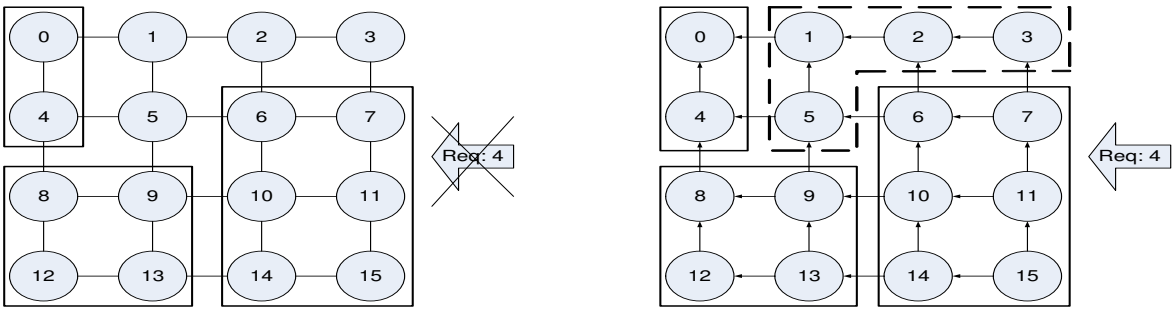
Fault-tolerance is also becoming a major concern when designing a NoC. As the integration of large numbers of components is pushed to smaller scales, a number of communication reliability issues are raised. Crosstalk, power supply noise, electromagnetic and inter-symbol interference are examples of such issues. Moreover, fabrication faults may appear, in the form of defective core nodes, wires or switches. In some cases, even though some regions of the chip are defective, the remaining chip area may be fully functional. From a NoC point of view, the presence of fabrication defects can turn an initial regular topology into an irregular one. In an off-chip network, the defective component(s) can simply be replaced, while for a multicore chip, if the routing layer cannot handle the fault(s), the chip is discarded. This is an issue known as yield and has a strong correlation to manufacturing costs.

Power consumption is also an issue of major concern. As buffers consume both area and energy, a NoC designer must strive to find a balance between acceptable power requirements and the need for buffers. To date, efforts have been made to minimize buffers both in size and number. Another issue related to power is that as the number of cores increases, cores may be under-utilized as applications may not require all the processing logic available on the chip (multimedia extensions for example) and the chip may simply have large idle times for particular applications. An example of the industry response to power consumption is Intel's recent introduction of a new state in their Penryn line of chips that allows the chip to put cores into a deep sleep state where the caches and cores are totally switched off. This points to the fact that mechanisms that dynamically reduce power are both mainstream and will be important in future multicore chips.

A number of challenging problems are emerging from the increased integration of cores on a chip. When designing a NoC for multicore systems, the three key metrics must be optimized at the same time. In this paper we explore viable solutions to all of them. We propose and argue how the use of virtualization techniques at the NoC level can help deal with the problems discussed above. From a conceptual point of view, a virtualized system provides mechanisms to assign disjoint sets of resources to different tasks or applications. Virtualization is valuable on a multicore chip since it allows for the optimization of the system along the three key design space directions (area/power, performance, yield). For instance, failed or idle components (that can be put to sleep) can be isolated within a proper virtualized configuration, thus providing the potential for higher yields and reductions in power consumption. A virtualized system can separate traffic belonging to different applications or tasks, preventing interference between them, leading to an overall higher performance.

The intention behind this paper is for it to serve as a position paper on the topic of virtualization for NoC and the challenges that should be met at the routing layer in order to maximize performance, fault-tolerance and power consumption.

The remainder of the paper is organized as follows. First, in Section 2 we introduce the concept of virtualizing a NoC and describe the advantages of such an approach in terms of performance, fault-tolerance and



(a) Resource allocations must be sub-mesh shaped, and a request for 4 resource entities are rejected even if 4 resource entities are free (external fragmentation).

(b) Resource allocations must constitute valid Up*/Down* sub-graphs, hence the request for 4 resource entities can be granted (broken line).

FIG. 2.1. *Three tasks are running when another request for resources arrives in a system consisting of 16 resource entities connected in a* 4×4 *mesh topology: The first task runs on entities number 0 and 4; the second task runs on entities number 6, 7, 10, 11, 14, and 15; and the third task runs on entities 8, 9, 12, and 13.*

power consumption. Then, in Section 3 we provide rough estimations on the impact of a virtualized NoC where we show how a virtualized NoC compares to a non-virtualized one. We conclude in Section 4 with a summary of our findings and discuss future work.

2. Virtualized NoC. The concept of virtualization is not a new one. It has been applied to different domains for different purposes. A few examples are virtual machines, virtual memory, storage virtualization, and virtual servers in data centers. A virtualized NoC may be viewed as a network that partitions itself into different regions, with each region serving different applications and traffic flows.

Many questions with regard to the design of a virtualized NoC can be posed. Does the virtualized system allow for the merging of different regions? Does the virtualized system support non-regular regions? How is traffic routed within regions? We believe that, in order to meet the design challenges with respect to performance, fault-tolerance, and power consumption, the most important issues for a virtualized NoC solution are: increasing resource utilization; the use of coherency domains; increasing the yield of chips; and reducing power consumption. In this section we provide an overview of each of these issues.

2.1. Increasing Resource Utilization. Current applications may not provide enough parallelism to feed all the cores that are available on a multicore chip. In such situations some of the cores may be under-utilized for periods of time, with the implication that resources are being wasted. In this situation, different applications (or tasks) should be allowed to use separate sets of cores concurrently. Hence, we need a partitioning mechanism to manage the resources and assign them to the applications in an efficient way.

A partitioning mechanism should, at least, meet the following two challenges, the minimization of fragmentation (maximization of system utilization) and the prevention of interference between messages (packets) that belong to different tasks. The partitioning of a multicore chip can be compared to the traditional processor allocation techniques, and some of the algorithms suggested for processor allocation could also be used for partitioning within a multicore chip. Contiguous allocation strategies (such as [6, 7, 8, 9, 10, 11, 12, 13]) designate a set of adjacent resources for a task. For meshes and tori most of the traditional contiguous strategies allocate strict sub-meshes or sub-cubes and cannot prevent the occurrence of high levels of fragmentation. External fragmentation is an inherent issue for contiguous strategies and occurs when a sufficient number of resource entities are available, but the allocation attempt nevertheless fails due to some restrictions (such as the requirement that a region of available resource entities must constitute a sub-mesh).

Some contiguous resource allocation algorithms have an attractive quality that we refer to as *routing-containment:* the set of resources assigned to a task is selected in accordance with the underlying routing function, such that no links are shared between messages that belong to different tasks. Routing-containment in resource allocation is important for a series of reasons. Most importantly, each task should be guaranteed a fraction of the interconnect capacity regardless of the properties of concurrent tasks. Thus, if one task introduces severe congestion within the interconnection network, other tasks should not be affected. In Section 3 we discuss results that show the effects of traffic overlapping. In earlier works routing-containment was an issue that was

often only hinted at. Even so, many strategies, like those that allocate sub-meshes in meshes, will be routing-contained whenever the DOR routing algorithm is used.

A close coupling between the resource allocation and routing strategies allows for the development of new approaches to routing-contained virtualization with minimal fragmentation. These approaches involve the assignment of irregularly shaped regions and the use of a topology agnostic routing algorithm. In general, these approaches can be used for any topology – a particularly attractive property in the face of faults in regular topologies.

One approach which is particular to the Up*/Down* [16] routing algorithm assumes that an Up*/Down* graph has been constructed for an interconnection network that connects a number of resources. For each incoming task the assigned set of resources must form a separate Up*/Down* sub-graph. This ensures high resource utilization (low fragmentation) since allocated regions may be of arbitrary shapes. In this case routing-containment is ensured without the need for reconfiguration. UDFlex [18] (for which performance plots that demonstrate the advantages of reducing fragmentation are included in Section 3) is an example of such a strategy.

Figure 2.1(a) shows a situation where three processes have been allocated to a NoC that does not support irregular regions. An incoming process that requires the use of four cores has to be rejected as the NoC only supports regions of regular shapes. Figure 2.1(b) on the other hand shows the case where a NoC supports the use of irregular regions. In the first case the DOR routing algorithm is used. Each packet must traverse first the X dimension and then the Y dimension. However, the acceptance of new applications is highly restricted as regions must be rectangular to allow traffic isolation with DOR routing (in Figure 2.1(a) node 5 could not communicate with nodes 2 and 3 without interfering with an already established domain). In Figure 2.1(b), however, the topology-agnostic Up*/Down* routing algorithm is used (node 0 is the root of the Up*/Down* graph). This allows allocation of irregular regions, which increases resource utilization.

In general, topology agnostic routing algorithms are less efficient than topology specific routing algorithms (although, in some cases, they perform almost as well [14, 15, 17]). Thus, when allocating arbitrarily shaped contiguous regions, there is a trade-off between routing-containment and routing efficiency. This is the case both for NoCs and off-chip networks. Also, when allocating arbitrarily shaped regions, reconfiguration may be needed to ensure routing-containment for incoming tasks.

The routing algorithm used in a NoC should be flexible enough to allow for the formation of irregularly shaped regions. The tight constraints applied to multicore chips with respect to issues such as latency, power-consumption, and area limit the choice of strategies available for routing and resource allocation. The use of routing tables requires significant memory resources in switches. Source-based routing, on the other hand, contains the entire path of a packet in its header and allows for a quicker routing decision when compared to the table approach. NoCs that allow routing tables or source-based routing are flexible with respect to the applicability of topology agnostic routing and assignment of arbitrarily shaped regions. For such environments the approaches described above may be used without restrictions, and solutions targeted for off-chip environments may be adopted. For some NoCs, however, the use of both routing tables and source-based routing may be unacceptable or disadvantageous. This group of NoCs can instead use a logic-based routing scheme. For allocation of sub-meshes in a mesh topology, DOR (which can easily be implemented in logic) is a possible choice. More interestingly, under certain conditions, recent solutions such as the region-based routing concept [4] and LBDR mechanism [5] allow the use of topology-agnostic routing algorithms in semi-regular topologies with small memory requirements within switches.

2.2. Coherency Domains. Typically, in a multicore system the L2 cache is distributed among all cores. This is a result of the use of L2 private caches or having a shared L3 on-chip cache (e.g. Opteron with Barcelona core). Each core has a small L2 cache and a coherency protocol is used. As the number of cores within a chip increases several new problems arise. Upon a write or read miss access from a core to its L2 cache the coherency protocol is triggered and a set of actions are taken. One possible action is to invalidate any possible copy of the block in the remaining L2 caches. This is normally achieved by using a snoopy protocol. However, implementing a snoopy protocol in a 2D mesh is difficult as snoopy actions do not directly map on to a 2D mesh.

A possible solution is to implement a snoopy protocol with broadcast packets. Although broadcasting solves the problem of keeping the coherency, as the system increases in number of cores, the broadcast action becomes inefficient. For instance, in a 80-core system broadcasting an invalidation is inefficient if only two or three copies of the block are present in the system. The entire network is flooded with messages that are only directed at two or three cores (and they will probably be physically close to each other). Figure 2.2(a) shows an example

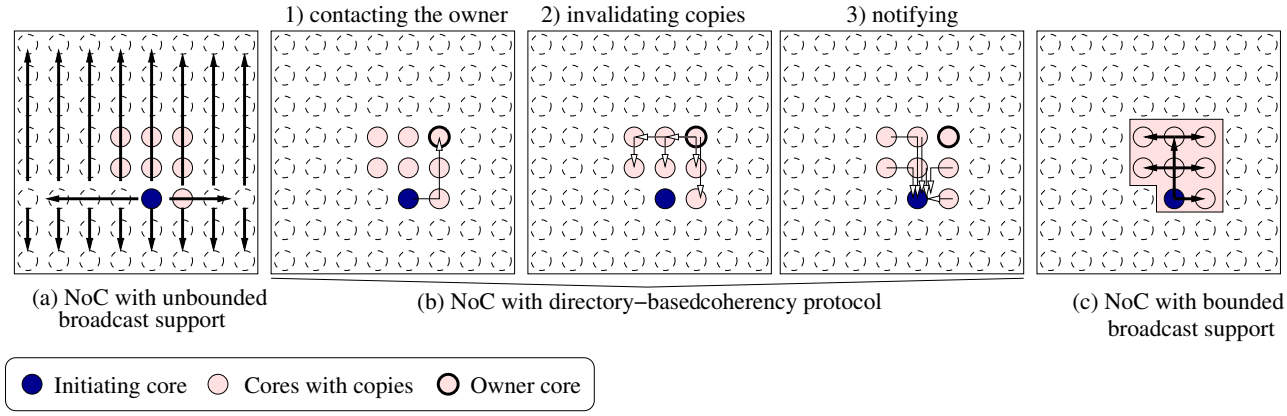


FIG. 2.2. *Examples of different coherency protocols.*

of this.

One possible solution is the use of directories to implement a coherency protocol. In this situation, each memory block has an assigned owner. Upon an action on a block the owner is inspected. The owner has a directory structure of all the cores that have copies of its block. Thus, the packet is sent only to the cores with copies of the accessed block. However, directories require significant memory resources that may be prohibitive for chip implementations as there are increases in latency due to indirection (see Figure 2.2(b)).

One solution that is becoming a reality is the use of coherency domains. When an application is started on a multicore system different threads or processes are launched on different cores and a possible solution is to map groups of cores that share data in a unique coherency domain. By implementing the required resources at the network and application level, the coherency protocol could remain enclosed within the coherency boundaries and thus prevent interference with traffic in other parts of the chip. Additionally, snoopy actions should be enclosed within the coherency domain by a restricted broadcast. In Figure 2.2(c) the snoopy action (implemented as a broadcast) is bounded to the coherency domain.

It is clear that within the context of virtualization for NoC, mechanisms that allow the use of localized broadcast actions within a coherency domain (network region) should be implemented. Additionally, it may be possible that some resources within the chip are shared by different domains, an example of this is memory controllers. In this situation the virtualized NoC must provide solutions to bound the traffic of each domain within its limits. This is required also for broadcast packets so each domain does not flood all the shared domains.

2.3. Yield and Connectivity of Chips. The manufacturing process of modern day chips consists of many stages but involves photo-lithographic processing of multiple logical as well as physical layers. Many factors can influence the quality of a chip and as high volume manufacturing technology scales from its current 45 nm to 32 nm and below, the occurrence of tile-based interconnect faults may become a greater issue to yield. Yield is defined as the number of usable chips versus defective chips that come off a production line, and it is expected that, as we go deeper into sub-micron processor manufacture, we will see decreases in yield [20].

A modern CPU consists of multiple metal layers, and it is common to use at least 10 layers to make an interconnect for a single core chip. When discussing yield and interconnects, it is important to make the distinction between the interconnect that makes up a single core within a chip (existing between multiple metal layers) and the global routing interconnect that is used between tiles.

Yield as a connectivity issue can benefit greatly from virtualization. Recall that once a chip has been manufactured, faulty components cannot simply be swapped out. NoCs need to be able to sustain faults that arise during production. Figure 2.3 shows two different fault situations. In the first case an entire block of the chip has been disabled as a result of a manufacturing defect. This chip can be recovered by the definition of a virtualized NoC with cores being grouped in a network region.

In the second case, however, a switch is disabled at the middle of the NoC. Two solutions arise here. First, the failed component can be excluded by a smart partitioning scheme that partitions the NoC into different regions. In a second solution, some simple static fault-tolerance mechanisms can be added to switches to

circumvent the failure within a region. We expect that solutions to yield can support one component failure.

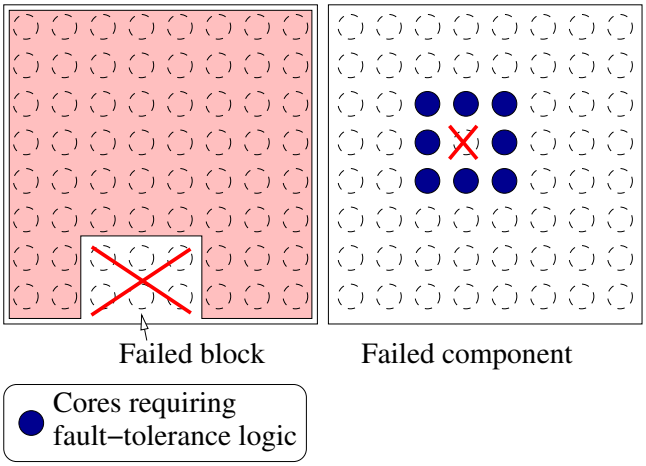


FIG. 2.3. *Dealing with component failures.*

An analysis of yield and connectivity was carried out in [19]. Two straight forward approaches to increasing yield as a result of faults was examined. These methods are shown in Figure 2.4.

The first, **D**imension-**O**rder **R**outing that supports **F**aults (DOR-F) that ensures continued connectivity when a fault is detected, is a mechanism used by the BlueGene/L torus network [22, 21]. This approach requires that the routers within each tile of the disabled area are prohibited from forwarding or accepting traffic that have the tile as destination or source. Rather it can solely act as a transit point forwarding traffic according to the DOR-YX algorithm. Hence the processing nodes are unreachable and can be switched off. As an example, with reference to Figure 2.4(a), tile 12 has been discovered to be faulty so tile 7 only forwards in the horizontal dimension. Similarly tile 13 only forwards in the vertical dimension. By doing so the remaining tiles within the NoC are guaranteed connectivity.

As DOR routing algorithm is still in use, no new turns are introduced to the channel dependency graph (according to the original DOR-YX) and hence the network retains its deadlock freedom. This method is referred to as DOR-F.

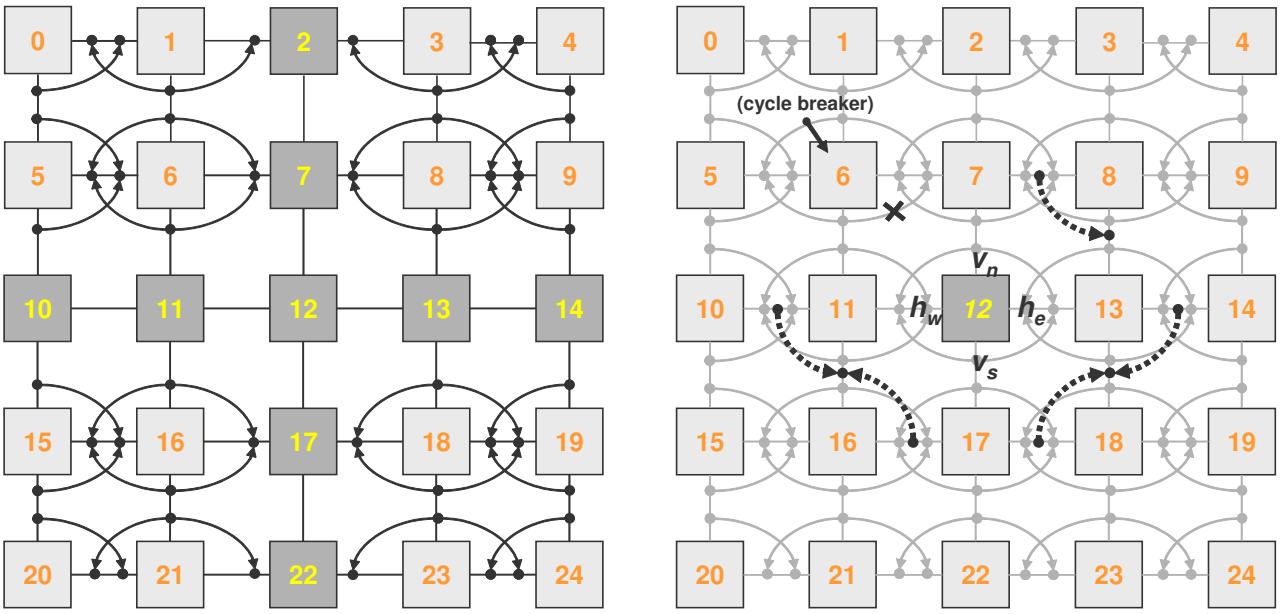
Another approach to the problem is based on an analysis based on **C**hannel **D**ependency **G**raphs that support **F**aults (CDG-F). The CDG-F method allows for DOR-YX routing in the fault-free case and when a packet's route does not cross a fault, but in the case where a packet crosses a fault it uses a re-routing mechanism. Figure 2.4(b) shows an example where tile 12 is faulty, and the five dependencies that must be added and the single dependency that must be removed in order to maintain connectivity and freedom from deadlock.

2.4. Reducing Power Consumption. Power consumption is a major concern. There are potential savings that can be made in effective power-aware routing techniques, that optimize the application traffic to ensure it finishes as quickly as possible and allow the network and cores to shut-down inactive parts.

The literature reports the interconnect power consumption at approximately 30% to 40% of total chip power consumption, a figure that also relates to the TeraScale chip. One reason the interconnect power consumption rate is so high is that the processing cores are not logic intensive (they are not fully fledged x86 cores). If this ratio was to significantly decrease then power aware routing may be less interesting.

Virtualization is also a power issue. In Section 2.1 we discussed issues relating to the sub-optimal allocation of processes to cores. If the operating system has tasks waiting to be allocated and is unable to allocate the tasks to the resources because of fragmentation, then the sub-optimal allocation strategy is not power-efficient and the operating system should ensure (fragmented) processing resources are put to sleep.

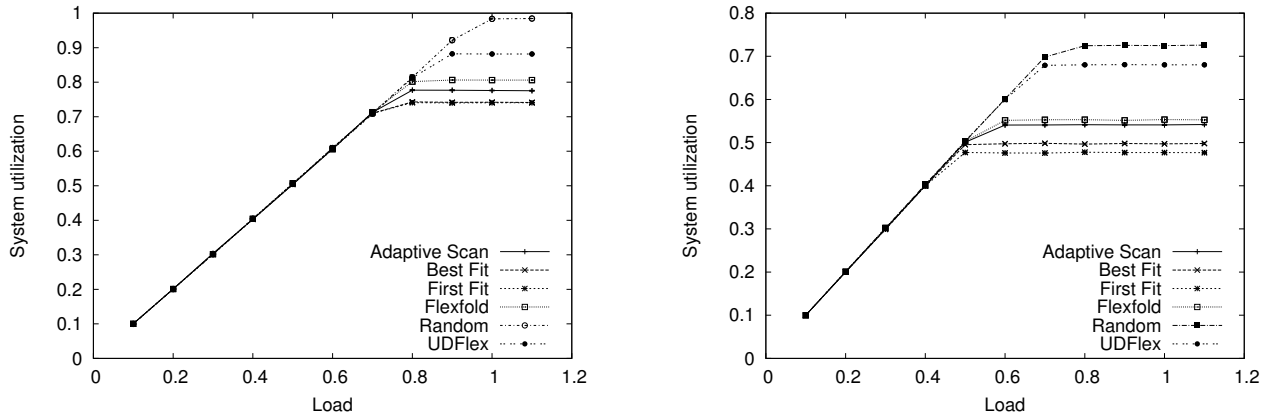
Virtualization also provides solutions to the power issue for NoC in the same way that it supports increases in yield by supporting fault-tolerance. Cores and interconnect components that are otherwise idling can be put to sleep and excluded at the NoC level by using the concept of virtualization, but in this setting, the virtualization technique has to be dynamic in nature, while from the yield point of view it could be static.



(a) The DOR-F. All switches/routers in the same row / column as switch 12 can only transit traffic

(b) The CDG-F method. New dependencies added to route deadlock free around switch 12

FIG. 2.4. *Illustrating two straight forward approaches that can help the yield/connectivity issue. Switch 12 is deemed faulty*



(a) Maximum task size is 16.

(b) Maximum task size is 256.

FIG. 3.1. *Effect on fragmentation (1 − system utilization) when allocating irregular regions. System utilization for a* 16×16 *mesh.*

3. Basic Examples. This section provides a basic set of examples with performance analysis to motivate virtualization of a NoC. First, we focus on the reduction of fragmentation effects when allowing allocation of irregular regions. Later, we evaluate the effects on performance when using traffic isolation through a virtualized NoC, either using unicast and multicast traffic. Finally, we analyze the yield improvement when using different routing algorithms.

3.1. Reducing Fragmentation Effects. Section 2.1 explained the fragmentation issue related to allocating contiguous regions of resources. Restricting allocations to regions of a particular shape (such as sub-meshes) results in severe fragmentation and, thus, low resource utilization. We argued that allowing irregularly shaped regions may reduce fragmentation, and introduced an approach, called UDFlex, that allows allocation of irregular regions (by allocating Up*/Down* sub-graphs). In Figure 3.1 the system utilization of UDFlex is compared with that of several strategies that allocate sub-meshes (Adaptive Scan, Flexfold, Best Fit and First Fit) for a

16×16 mesh topology. Random is a non-contiguous fragmentation-free strategy, and, thus, a theoretical upper bound performance indicator with respect to system utilization.

Each task requests $a \times b$ resource entities (which is considered a product of numbers for the strategies that allow allocation of irregular regions –UDFlex and Random– and a sub-mesh specification for the other strategies). a and b are drawn from separate uniform distributions with maximum sizes a_{max} and b_{max}, respectively. Figures 3.1(a) and 3.1(b) show system utilization for tasks with $\{a_{max} = 4, b_{max} = 4\}$ and $\{a_{max} = 16, b_{max} = 16\}$, respectively. In both cases the increased flexibility of allocating irregular regions reduces fragmentation and increases the utilization of system resources when compared to sub-mesh allocation. We note that the advantage of allocating irregular regions is even more pronounced for the larger task sizes. A study of the torus topology gave similar results as for mesh.

3.2. Performance Impact From Traffic Isolation: Unicast Packets. Traffic isolation is the main property a virtualized system must achieve. In such a situation, traffic from one domain is not allowed to traverse other domains. To see the performance effects of a non-isolated traffic situation, we compare different routing algorithms and their behavior in a 8×8 mesh split into two domains (see Figure 3.2).

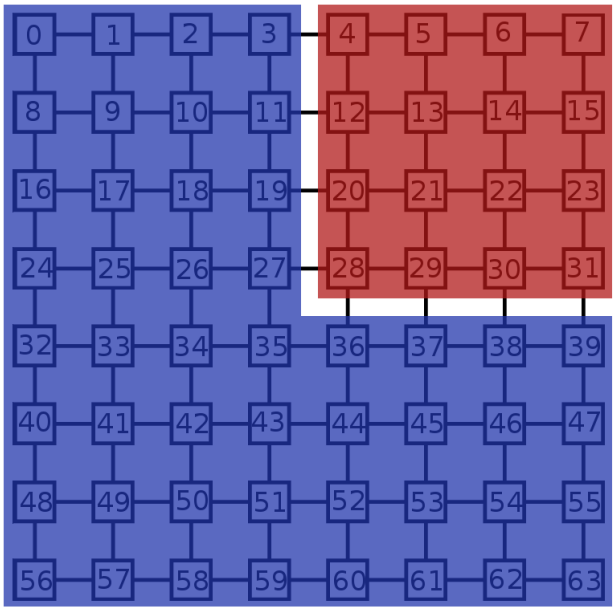
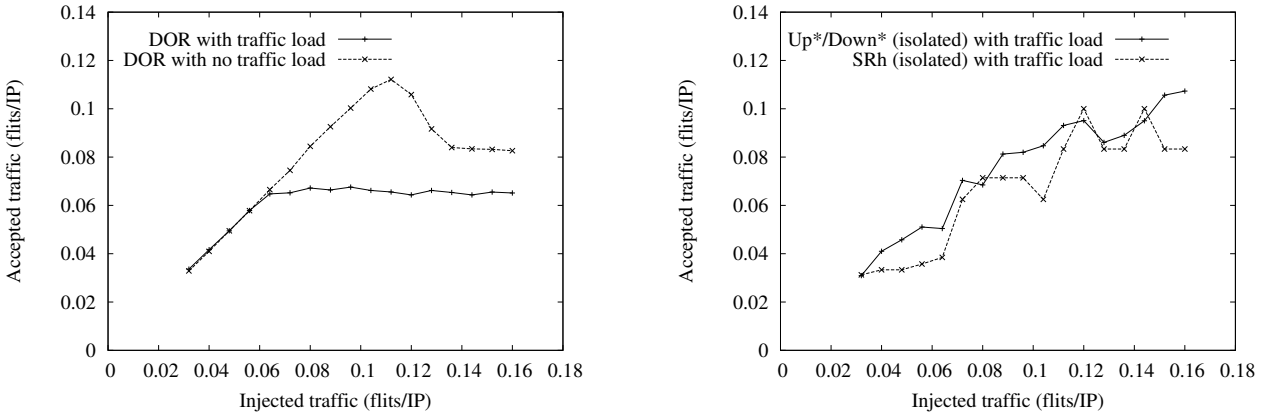


FIG. 3.2. *Domains evaluated for traffic isolation on a* 8×8 *mesh.*

The first domain consists of the top-right 4×4 sub-mesh. This domain is tested with no traffic load (this is plotted as "no load" in the figures) and with a congested situation (this is plotted as "with load" in the figures). The remaining network belongs to a second domain where we inject the full range of traffic from low load to saturation. We have evaluated the following routing algorithms: *DOR*, SR_h and *Up*/Down**. Recall that *DOR* can not sustain routing containment in an irregular topology, so it can not prevent interference between both domains. For SR_h and *Up*/Down**, however, traffic is isolated within the domains. In all the evaluations wormhole switching is assumed.

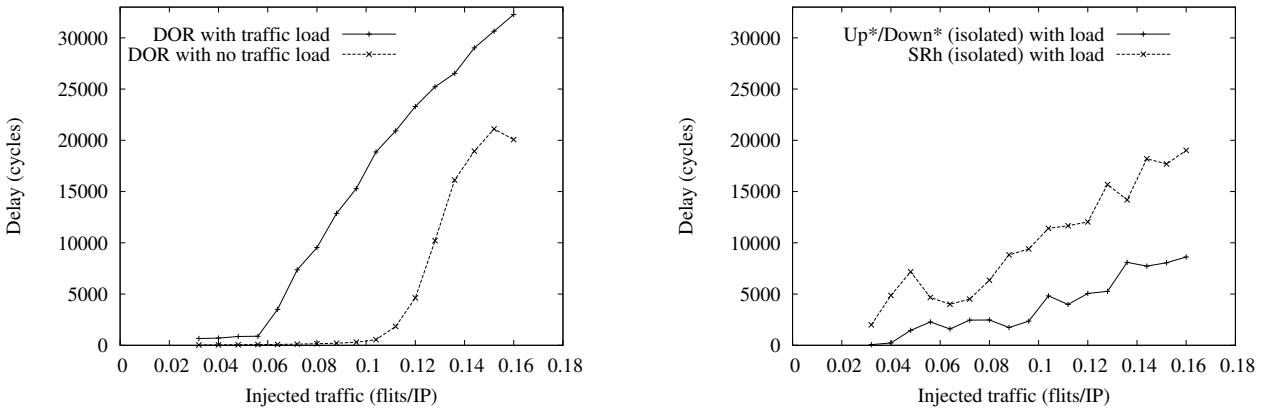
In Figure 3.3.a we can observe the effects on performance when traffic isolation is not enforced by the routing algorithm. When using *DOR* there are many packets that must traverse the congested domain (top-right 4×4 sub-mesh). As can be seen, network throughput dramatically drops to 0.07 flits/IP. In the absence of traffic in the sub-mesh, network throughput would achieve 0.11 flits/IP, instead. Note that SR_h and *Up*/Down** (see Figure 3.3.b) which enforce traffic containment do achieve such throughput numbers even in the presence of congestion in the sub-mesh. In an isolated system, packets do not cross domains and are thus not affected by the congested traffic.

Delay is also observed in Figure 3.4 to be significantly greater when traffic is routed with *DOR*. When the small domain is congested, average packet latency for the large domain is one order of magnitude higher (when



(a) No traffic isolation with *DOR*. (b) Traffic isolation with SR_h and *Up*/Down**.

FIG. 3.3. *Performance impact from traffic isolation. Network throughput for an* 8×8 *mesh.*



(a) No traffic isolation with *DOR*. (b) Traffic isolation with SR_h and *Up*/Down**.

FIG. 3.4. *Performance impact from traffic isolation. Network latency for an* 8×8 *mesh.*

compared with the case that the small domain has no traffic). On the other hand, packet latency is much lower for SR_h and *Up*/Down**. With *DOR*, every packet that crosses the congested domain also gets congested, thus extending the congestion to both domains.

3.3. Broadcast and Coherency Domains Analysis. As previously stated, it is desirable for a virtualized system to provide a broadcast mechanism bounded to the domains in which the NoC is partitioned. Unbounded broadcast actions affect the overall network performance and thus may be prone to congestion and undesirable effects (mainly increased packet latency).

For this evaluation, two configurations were used. In the first configuration the entire NoC, a 8×8 mesh, is used with no domain definition, therefore, a broadcast floods the entire network. In the second configuration, however, the mesh is divided into four domains as depicted in Figure 3.5, and a broadcast is initiated within each domain. In the latter case, each broadcast only floods the domain where it was issued. The four broadcasts are issued at the same time from different nodes (9, 30, 36 and 59). The performance has been evaluated both with no background traffic present in the network and with unicast background traffic, randomly distributed within its domain. In each scenario the broadcast latency is measured from the time the broadcast is initiated to the time the last end-node received the message header (i. e. all broadcast actions are finished).

Figure 3.6 shows the results. The first and second columns show the broadcast latency for both the configurations under study when no background traffic is injected. The third and fourth columns show results for both configurations with background traffic. As can be observed, when uniform background traffic is

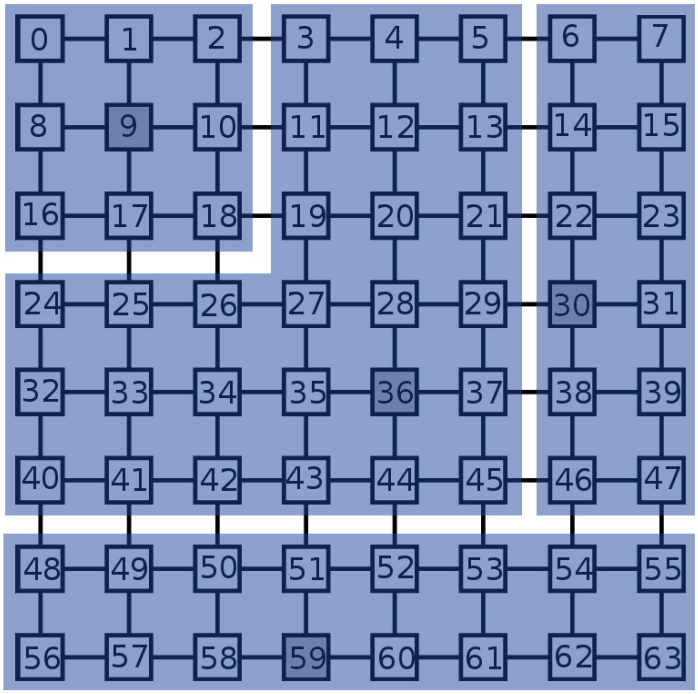


FIG. 3.5. *A* 8×8 *mesh divided in 4 domains for bounded broadcast performance evaluation.*

present in the network, using broadcast domains significantly improves the overall performance of the network. In particular, when broadcast domains are used, 239 cycles are required to handle the four broadcasts, whereas 17273 cycles are needed when no broadcast boundaries are used.

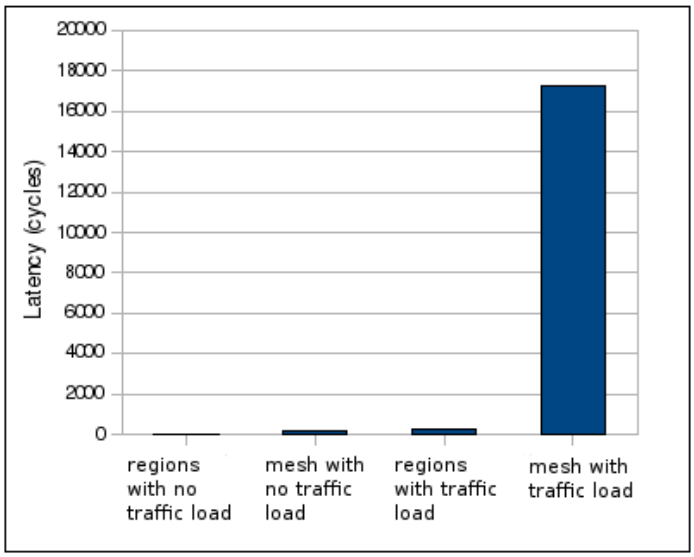


FIG. 3.6. *Effect on latency in a* 8×8 *mesh with and without broadcast domains.*

3.4. Yield and Connectivity Analysis. The potential improvements in yield by taking yield into account at the routing level is obtained by increasing connectivity for a NoC with faults and thus the number of usable / routable cores within a NoC. When analyzing yield and connectivity it is important that the chip remains connected and deadlock free. If neither of these conditions are met then we assume that the chip is defective.

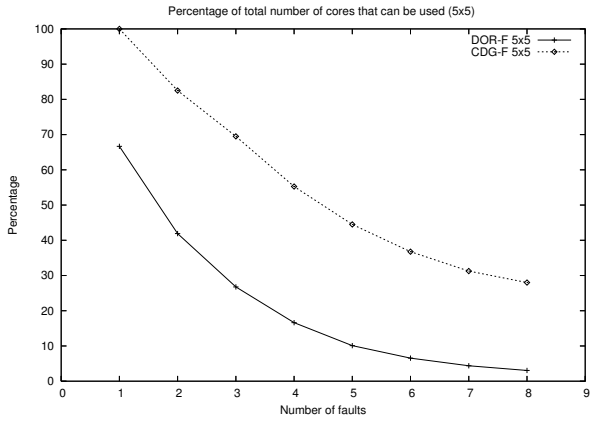
Both the CDG-F and DOR-F methods were tested. For the CDG-F method, an initial channel dependency graph for the given mesh size using DOR-YX routing, without any faults is computed. The desired number of faults is drawn from a random distribution, and for each fault new channel dependencies are added and removed as described in Figure 2.4(b). Dependencies are added to the channel dependency graph as long as it does not introduce a cycle. Once a cycle is found, the chip is assumed to be deadlock-prone and unusable.

Next, a routing-connectivity check is run and involves checking the path from each source to each destination within the chip. The computation of routing paths is based on the channel dependency graph and details whether or not the routing function is connected for a given set of faults. Note that even though there may be physical connectivity on the chip there may not be routing-connectivity. This can be a result of turns that may result in deadlock if such new turns are introduced. We only consider chips where the tiles have physical connectivity.

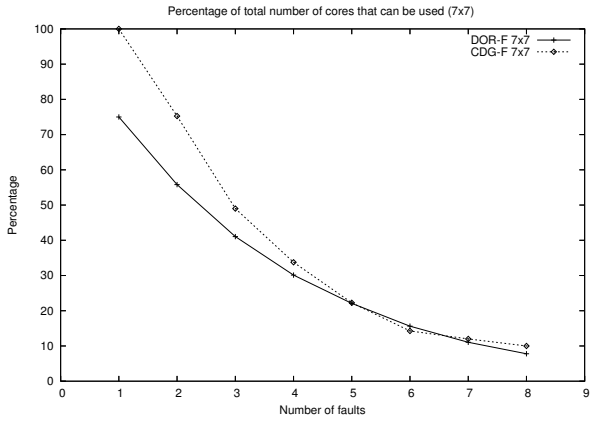
For the DOR-F method the connectivity is analyzed as described in Figure 2.4(a). The DOR-F method will always maintain routing connectivity at an extreme cost associated with switching off processing nodes within tiles.

We analyzed the difference between the two methods (DOR-F and CDG-F). The goal was to draw an over-all picture of the yield situation to see which of the two methods provide the overall highest number of routable (usable) tiles. The answer to this question is shown in Figure 3.7.

Figure 3.7 shows the percentage of tiles that are connected and thus usable / routable for mesh sizes 5×5 and 7×7. Common amongst both plots is the fact that the CDG-F method out-performs the DOR-F for a single fault. With smaller network sizes, CDG-F outperforms DOR-F by a significant margin, and we see the performance cross over point, where both methods exhibit the same performance, in the 7×7 mesh after the occurrence of five faults. For increasing sizes of networks, greater than 7×7, the DOR-F method outperforms the CDG-F method after two faults.



(a) 5×5. CDG-F outperforms DOR-F method



(b) 7×7. Showing the performance cross over point for both methods

FIG. 3.7. *The percentage of connected and thus usable cores plotted against an increasing number of faults.*

These graphs represent the connectivity yield of chips against an increasing number of faults and may be of interest to high volume NoC manufacturers when deciding to adopt a given yield-aware routing strategy.

Manufacturers may already be in a position to anticipate a product-line of certain sizes and ranges but it is unlikely that the exact details of how larger multicore chips will be manufactured has yet been decided.

For situations where only ports within a router have faults, then the interconnect failure can be seen as a link failure, and [23] have detailed how link failures can be handled in mesh networks.

4. Conclusion. Networks on Chip is still a new research area, and many problems are still to be solved. In this paper we have concentrated on three central problem areas, namely power consumption, increasing production yield by utilizing the functional parts of faulty chips, and performance as a measurement of resource utilization. Our position is that these problems can be resolved using virtualization techniques.

The research that is needed to leverage the potential of NoC will be across several fronts: New and flexible routing algorithms that support virtualization need to be developed. We demonstrated how DOR and

Up*/Down* have very different properties with respect to fragmentation under the requirement of routing containment, and we believe that routing algorithms specifically designed for this purpose will be beneficial.

Furthermore, the concept of fault tolerance in on-chip interconnection networks will take on another flavor, as permanent faults on a chip need to be handled with mechanisms that require very little area. Mechanisms that need complex protocols or extra routing tables are therefore of limited interest.

Finally, whereas earlier work on power aware routing has focused on turning off or reducing the bandwidth of links as traffic is reduced, we are now facing a situation where several of the cores on the chip may be turned off. This is a new situation in the sense that where previous methods needed to react to traffic fluctuations over which they had very limited control, we can now develop methods for power-aware routing under the assumption that, to some extent, we can control which cores are to be turned off. This gives an added flexibility in the way we handle the inherent planning of problems with sleep/wake-up cycles.

The intention of this paper is not to provide complete solutions to the problems we address. Rather, we have explained the properties of these problems, quantified some of them through simulations, and explained and demonstrated how virtualization shows promise as a unifying solution to all of them.

REFERENCES

[1] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar: A 5-GHz Mesh Interconnect for a Teraflops Processor. In *IEEE Micro Magazine*, Sept-Oct. 2007, pp. 51-61.

[2] J. A. Kahle, M. N. Day, H. P. Hofstee, C. R. Johns, T. R. Maeurer, and D. Shippy: Introduction to the Cell multiprocessor. *IBM Journal of Research and Development*, vol. 49, no. 4/5, 2005.

[3] InfiniBand Trade Association: *InfiniBand Architecture Specification version 1.2.* `http://www.infinibandta.org/specs` 2004.

[4] J. Flich, A. Mejía, P. López, and J. Duato: Region-Based Routing: An Efficient Routing Mechanism to Tackle Unreliable Hardware in Networks on Chip. In *First International Symposium on Networks on Chip (ISNOC)*, 2007.

[5] J. Flich, S. Rodrigo, and J. Duato: An Efficient Implementation of Distributed Routing Algorithms for NoCs In *Second International Symposium on Network-on-Chip (ISNOC)*, 2008.

[6] J. Ding and L. N. Bhuyan: An Adaptive Submesh Allocation Strategy for Two Dimensional Mesh Connected Systems. In *International Conference on Parallel Processing (ICPP'93)*, page 193, 1993.

[7] V. Gupta and A. Jayendran: A Flexible Processor Allocation Strategy for Mesh Connected Parallel Systems. In *International Conference on Parallel Processing (ICPP'96)*, page 166, 1996.

[8] M. Kang, C. Yu, H. Y. Youn, B. Lee, and M. Kim: Isomorphic Strategy for Processor Allocation in k-ary n-cube Systems. *IEEE Transactions on Computers*, 52(5):645–657, May 2003.

[9] F. Wu, C.-C. Hsu, and L.-P. Chou: Processor Allocation in the Mesh Multiprocessors Using the Leapfrog Method. *IEEE Transactions on Parallel and Distributed Systems*, 14(3): 276–289, March 2003.

[10] Y. Zhu: Efficient Processor Allocation Strategies for Mesh-Connected Parallel Computers. *Journal of Parallel and Distributed Computing*, 16(4): 328–337, Dec 1992.

[11] P.-J. Chuang and C.-M. Wu: An Efficient Recognition-Complete Processor Allocation Strategy for k-ary n-cube Multiprocessors. *IEEE Transactions on Parallel and Distributed Systems*, 11(5): 485–490, May 2000.

[12] W. Mao, J. Chen, and W. Watson III: Efficient Subtorus Processor Allocation in a Multi-Dimensional Torus. In *8th International Conference on High-Performance Computing in Asia-Pacific Region*, page 53, 2005.

[13] H. Choo, S.-M. Yoo, and H. Y. Youn: Processor Scheduling and Allocation for 3D Torus Multicomputer systems. *IEEE Transactions on Parallel and Distributed Systems*, 11(5):475–484, May 2000.

[14] O. Lysne, T. Skeie, S.-A. Reinemo, and I. Theiss. Layered Routing in Irregular Networks. *IEEE Transactions on Parallel and Distributed Systems*, 17(1):51–65, January 2006.

[15] J. C. Sancho, A. Robles, J. Flich, P. López, and J. Duato: Effective Methodology for Deadlock-Free Minimal Routing in InfiniBand Networks. In *International Conference on Parallel Processing (ICPP'02)*, pages 409–418, 2002.

[16] M. D. Schroeder, A. D. Birrell, M. Burrows, H. Murray, R. M. Needham, T. L. Rodeheffer, E. H. Satterthwaite, and C. P. Thacker: Autonet: A High-Speed, Self-Configuring Local Area Network Using Point-to-Point Links. SRC Res. Rep. 59, Digital Equipment Corp., 1990.

[17] T. Skeie, O. Lysne, J. Flich, P. López, A. Robles, and J. Duato: LASH-TOR: A Generic Transition-Oriented Routing Algorithm. In *International Conference on Parallel and Distributed Systems (ICPADS'04)*. IEEE, 2004.

[18] Å. G. Solheim, O. Lysne, T. Sødring, T. Skeie, and J. A. Libak: Routing-Contained Virtualization Based on Up*/Down* Forwarding. In *International Conference on High Performance Computing (HiPC'07)*, pages 500–513, 2007.

[19] T. Sødring, Å. G. Solheim, T. Skeie, S.-A Reinemo: An Analysis of Connectivity and Yield for 2D Mesh Based NoC with Interconnect Router Failures. To appear in *11th EUROMICRO Conference on Digital System Design (DSD'08)*, 2008.

[20] N. Campregher, P. Y. K. Cheung, G. A. Constantinides, and M. Vasilko: Analysis of Yield Loss due to Random Photolithographic Defects in the Interconnect Structure of FPGAs. In *International Symposium on Field-Programmable Gate Arrays (FPGA'05)*, pages 138–148, 2005.

[21] M. E. Gómez, J. Duato, J. Flich, P. López, A. Robles, N. A. Nordbotten, O. Lysne, and T. Skeie: An Efficient Fault-Tolerant Routing Methodology for Meshes and Tori. In *IEEE Computer Architecture Letters*, vol. 3, no. 1, 2004.

[22] W. Barrett and The BlueGene/L Team: An Overview of The BlueGene/L Supercomputer. In *Proceedings of the 2002 ACM/IEEE Conference on Supercomputing*, CD ROM, 2002.

[23] O. Lysne, T. Skeie, and T. Waadeland: One-Fault Tolerance and Beyond in Wormhole Routed Meshes. In *Microprocessors and Microsystems*, 21(7/8):471–480, 1998.

Edited by: Sabri Pllana, Siegfried Benkner
Received: June 16, 2008
Accepted: July 28, 2008