Efficient Unicast and Multicast Support for CMPs

Samuel Rodrigo, José Flich, and José Duato
Parallel Architectures Group
Technical University of Valencia, Spain
e-mail: srodrigo@gap.upv.es

Mark Hummel
AMD fellow
AMD
e-mail: Mark.Hummel@amd.com

Abstract

Beyond a certain number of cores, multi-core processing chips will require a network-on-chip (NoC) to interconnect the cores and overcome the limitations of a bus. NoCs must be carefully designed to meet constraints like power consumption, area, and ultra low latencies. Although 2D meshes with DOR (Dimension-Order-Routing) meet these constraints, the need for partitioning (e.g. virtual machines, coherency domains) and traffic isolation may prevent the use of DOR routing. Also, core heterogeneity and manufacturing and run-time faults may lead to partially irregular topologies. Routing in these topologies is complex, and previously proposed solutions required routing tables, which drastically increase power consumption, area, and latency. The exception is LBDR (Logic-Based Distributed Routing), a flexible routing method for irregular topologies that removes the need for using routing tables (both at end-nodes and switches), thus achieving large savings in chip area and power consumption. But LBDR lacks support for multicast and broadcast, which are required to efficiently support cache coherence protocols both for single and multiple coherence domains.

In this paper we propose bLBDR, an efficient multicast and broadcast mechanism built on top of LBDR. bLBDR performs multicast operations using a logic-based broadcast within a domain (a region with bounds). This allows us to isolate the traffic into different domains, thus enabling the concept of virtualization at the NoC level. Also, bLBDR extends the concept of routing regions in LBDR by providing a mechanism that allows the flexible definition of multiple domains, sets of network resources.

bLBDR fulfills all the practical requirements, including not only low latency and power and area efficiency, but also support for virtualization, partitionability, fault-tolerance, traffic isolation and broadcast across the entire network as well as constrained to coherency domains or regions. All this is achieved by a small and power efficient routing logic (7x area savings and 17x power reduction when compared to a routing table in an 8×8 mesh network).

1. Introduction

Multi-core architectures have become mainstream for designing processors for the embedded, desktop, and server markets. As designers found limits to improve the performance of single-core solutions, industry shifted to the multi-core paradigm where multiple and possibly simpler processor cores are integrated into the same chip. Although the number of cores in current processing devices is rather small (i.e. two to eight cores per chip), this trend is expected to change (e.g. Teraflop research chip with 80 cores [1]).

Chip architectures with such a large number of cores will require a high-performance network on chip (NoC) to efficiently interconnect cores among them and with cache blocks and/or memory controllers. Current chip implementations are based on bus or ring network structures (e.g. the Cell multiprocessor [2]). However, as the number of cores increases, such network topologies become the bottleneck of the system, as they do not scale. For chips with a large number of homogeneous cores, a 2D mesh topology is usually preferred due to its layout on a planar surface in the chip. This is the case for the Teraflop chip.

Besides the use of NoCs in chip multiprocessors (CMPs), there are other architectures that also benefit from a NoC. Examples are TRIPS [3], RAW [4] and Wavescalar [5] where innovative architectures based on operand networks are used. In this case, the NoC is used to transfer operands between functional units. The TRIPS operand network uses a 5-ary 2D mesh.

A large body of research has been undertaken on off-chip networks and so most mechanisms, techniques and methods can be applied to NoCs. However, new physical constraints appear in NoCs, which were not a primary concern in the off-chip domain. In particular, NoC designers must not overlook area, power and latency requirements. As an example, within the context of a NoC, ultra-low latencies are typically required, so every stage must be carefully optimized. This is the reason why logic-based routing is the preferred solution to route packets with minimum delay. In general, 2D meshes with DOR deliver solutions with very low latency, and power and area requirements.

However, fault-tolerance is becoming a major concern in NoCs and CMPs. Due to the high integration scale, a

978-1-4244-2837-3/08/$25.00 ©2008 IEEE

number of different communication reliability issues appear. Crosstalk, power supply noise, electromagnetic and inter-symbol interference are some of the problems to be considered. Moreover, manufacturing faults, like inoperative cores, wires or switches may also be present inside the chip. In all these cases, while some components of the chip are defective, the remaining area is still fully functional and, unlike in the off-chip domain, the defective components can not be replaced. The result is that we are dealing with an irregular topology instead of the initial regular one, and the routing layer must take this into account (this issue is related with yield and has strong correlation to manufacturing costs).

Multi-core chips open new paths to explore, too. In order to exploit the large number of cores inside, and due to the fact that applications are not exploiting enough parallelism, virtualization of the chip is becoming a necessity. In a virtualized system resources are distributed among different virtual machines. Although the virtualization concept is not new, applying it to NoCs and CMPs is challenging. The network must guarantee traffic isolation, thus leading to irregular sub-networks even within the original 2D mesh. Figure 3 shows an example where different regions are defined.

Recent works motivate virtualization at the IP level [7] as well as at the memory system level [10]. In this paper we go a step further by proposing virtualization at the NoC level. In particular, we propose a routing mechanism that aims at helping the implementation of a virtualized NoC for multi-core chips (either CMPs, operand-based architectures, or SoCs). Three main properties are envisioned for such on-chip network. First, the implementation of the routing algorithm needs to be compact and efficient (to meet the latency, area and power constraints in chip multiprocessors and operand networks) while providing support for irregular topologies. Second, a multicast mechanism must be provided to achieve faster and efficient collective communication, which is required by higher level entities like cache-coherent shared memory protocols (either directory-based [8] or token-based [9]), operand networks (when an operand is requested by several functional units), or new techniques (virtual hierarchies [10]). Finally, a compact mechanism to help define and isolate resources by defining multiple regions must be provided.

The first property (an efficient routing implementation for irregular topologies) has been recently addressed by the proposal of the Logic-Based Distributed Routing (LBDR) mechanism [12]. LBDR requires minimum logic (a set of flip-flops and logic gates per switch) and allows the use of different routing algorithms for irregular topologies, removing the need for using tables at switches (when using distributed routing) and end-nodes (when using source routing). In this paper we present the bLBDR mechanism. bLBDR extends LBDR functionality in two directions (providing the remaining two properties required for a virtualized NoC). First, by providing broadcast and multicast support within the NoC. Second, by defining regions and isolating the traffic within a given region.

Indeed, a key problem that needs to be addressed is the support for efficient multicast and broadcast in NoCs. In [11] it is demonstrated that multicast traffic is present in most of the environments where NoCs are needed. In particular, more than 5% of multicast traffic has been observed in directory-based protocols, token-based coherence protocols and operand networks. In [11] also, the impact of multicast traffic is analyzed, showing the slowdown in execution time, which reaches, in some cases, a factor of 2.2 when no multicast support is provided at all.

On the other hand, there are many benefits when enabling the definition of network regions. Defective components (due to manufacturing defects or problems related to the high-integration scale) can be tolerated with the use of network regions. Simply, defective elements are isolated by defining appropriate regions that cover the entire set of working elements. Power management is also critical in a NoC. The literature reports the interconnect power consumption at approximately 30% to 40% of total chip power consumption. Power-aware techniques must be defined to allow network components and cores to shut down inactive parts. Properly defining the network regions helps to put inactive resources (defined in different regions) to sleep.

The definition of network regions also promotes the use of coherency domains. On CMPs, different tasks or processes are mapped into different cores. It is likely that a task mapped on a group of cores needs to share data among them (typically across cache blocks) efficiently. The use of a coherency domain helps constraining the coherency protocol within the domain boundaries thus preventing interference with traffic from other regions of the chip. Also, if broadcast is allowed inside the domain much savings can be achieved.

Finally, application domains also benefit from NoC partitioning. An application domain is defined as the complete set of cores used by the application. The definition of these domains helps techniques that ensure minimal fragmentation (both external and internal) when applications are mapped onto a group of cores.

One of the key benefits of bLBDR is its simplicity to define regions and allow multicast and broadcast traffic with no routing table requirements at all, thus being suitable for NoCs. Although other solutions exist to provide broadcast and multicast support (see Section 6) they either need tables implemented at switches or end-nodes, or they do not allow irregular topologies or regions. Thus, they lack the support needed to enable a virtualized system at the NoC level.

The rest of the paper is organized as follows. In Section 2, the LBDR mechanism is briefly described. In Sections 3 and 4, bLBDR with region support and multicast/broadcast facilities is described. In Section 5 evaluation results are

provided. Section 6 analyzes related work. Finally, the paper is concluded with Section 7 where we draw some conclusions.

2. LBDR Description

bLBDR is built on top of LBDR (an efficient implementation of unicast distributed routing). For LBDR method to be applicable, two conditions must be fulfilled:

- Packets are routed with X and Y offsets.
- Every end-node must communicate with any other end-node through the use of minimal paths (non-minimal paths are not supported).

LBDR uses two sets of bits: the R_{xy} routing bits and the C_x connectivity bits, making a total of 12 bits per switch, 3 bits per output port (two R_{xy} bits and one C_x bit). The four output ports are labeled as N, E, W and S. A C_x bit defines the connectivity at the x output port. For example, if the C_n bit is set it means there is a neighbor switch connected through the N port. The R_{xy} bit at a given switch indicates if a packet is allowed (by the applied routing algorithm) to leave the switch through output port x and at the next switch to turn to direction y. For example, if R_{ne} bit is set, a packet can be routed in the current switch through the N output port and at the next switch through the E output port. With R_{xy} bits a 1-hop visibility of the allowed turns is provided.

The four connectivity bits (per switch) are Cn, Ce, Cw, and Cs, while the routing bits (per switch) are R_{ne}, R_{nw}, R_{en}, R_{es}, R_{wn}, R_{ws}, R_{se}, and R_{sw}. These bits are set before normal operation and are computed based on the routing algorithm used and the current topology. This can be easily achieved by representing the routing algorithm as a set of routing restrictions (see Figure 1). A routing restriction is set by two adjacent links and no packet can cross a routing restriction. Remember that the routing algorithm must meet the conditions of deadlock-freedom, connectivity, and minimal path support.

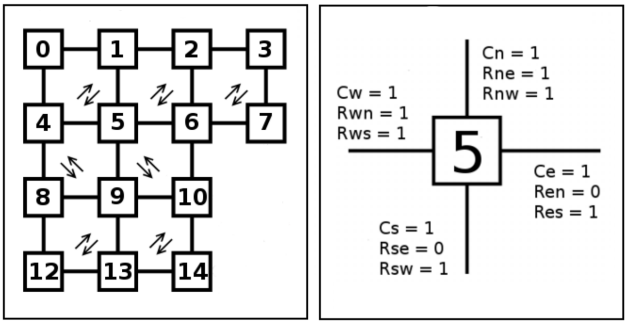

Figure 1. Routing algorithm represented by restrictions.

LBDR logic is composed of two units. Figure 2 shows the logic for the north port (similar logic is provided for each port). In the first unit, current X and Y coordinates of the switch are compared to X and Y packet's destination coordinates. This unit, thus, chooses the candidate output ports where a packet can be sent through. If destination coordinates are equal to current coordinates, the packet is routed to the local port (not shown).

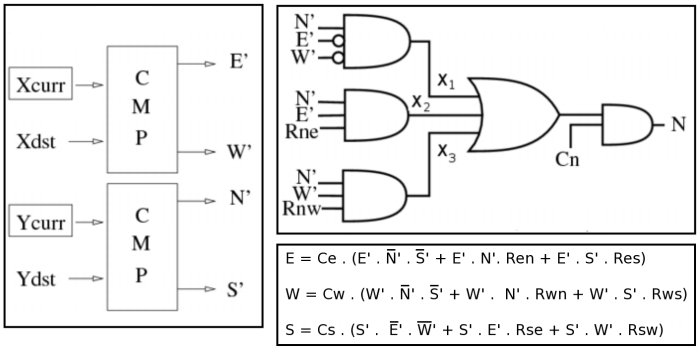

Figure 2. LBDR logic.

The second unit is the mechanism's core. First, the logic checks if the packet can be routed in each direction with the help of the R_{xy} routing bits. For example, a packet being routed through N output port may have any of three routing directions: north (signal x_1), north-east (signal x_2) and north-west (signal x_3). If going north-east, the N port will be provided only if a north-east change is allowed at the next switch (R_{ne} bit is set). Similar decisions are taken when forwarding the packet north-west (R_{nw} bit). Therefore, LBDR filters candidate output ports based on R_{xy} bits.

Finally, once the filter with R_{xy} bits is performed, LBDR applies a second filter by using the C_x bits. Simply, a candidate output port is eligible (by the scheduler) if there is connectivity through the output port (C_x bit is set).

LBDR allows many routing algorithms and irregular (and regular) topologies as long as deadlock-freedom and connectivity are guaranteed by the routing algorithm. For a detailed description of LBDR, please refer to [12].

3. bLBDR: Network Regions

The first improvement over the LBDR mechanism is the definition of networks regions or domains. In this section we describe how LBDR is upgraded to support regions and how regions can be configured.

Isolated network regions can be defined by using connectivity bits (C_x) available in LBDR. Figure 3 shows a virtualized NoC with 4 different network regions: two applications are mapped on different chip resources, one region is powered off and another one is marked as failed. For the two regions where an application is run connectivity bits are shown. As can be noticed, at the boundaries of each region connectivity bits are set to zero, thus preventing packets from leaving the region. As an example, for packets going from switch A to

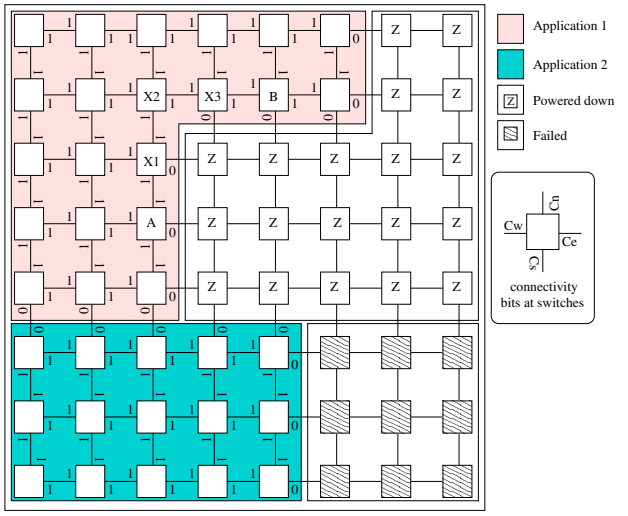

Figure 3. Disjoint regions and C_x bits.

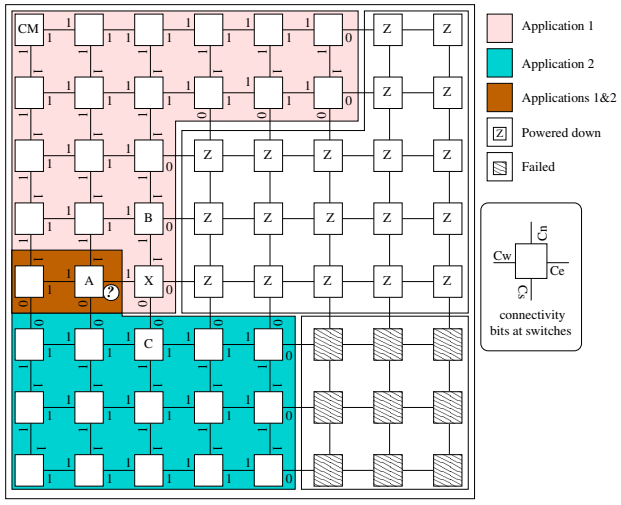

Figure 4. Overlapped regions and C_x bits.

switch B (in the figure) connectivity bits enforce the use of a path inside the region (going through switches $X1$, $X2$, and $X3$), thus avoiding the region with powered down switches.

However, it is highly interesting (and convenient) to support the definition of overlapping regions as well. Overlapped regions offer several advantages. For example, the possibility of sharing common resources (e.g. memory controllers, cache resources) between different applications or threads. Figure 4 shows the previous case but now both applications share some chip resources (two IP blocks). In this case, traffic in both regions must be allowed to access the shared region. However, using the connectivity bits provided by LBDR is not enough. For instance, the C_e connectivity bit at switch A can not be set with a common value for both regions. For application 1, the bit needs to be set to one (to allow paths from A to B passing through switch X). However, for application 2, the bit needs to be reset (to prevent packets from going outside the region, e.g. path A-X-C).

Another example where overlapping regions is required can be deduced from Figure 4. In this case, communication is required at the entire chip (e.g. control information). For instance, node CM (*chip manager*) may decide to power on some resources in order to allocate a new application inside the chip. Thus, data needs to be sent from node CM to other nodes outside the region where CM lies. In this case, connectivity bits used in LBDR can not help.

To support overlapped regions in bLBDR connectivity bits are extended for each possible overlapping domain. For instance, in order to support up to 8 overlapped regions, each LBDR connectivity bit is extended to eight bLBDR connectivity bits. As an example, connectivity through the N port is represented now by an 8-bit register (C_n) with $C_{n[0]}$ $\ldots$ $C_{n[7]}$ bits, each one indicating the connectivity through the north port for each possible region.

Packets need also to be labeled with the region they belong to. This can be done, basically, in two ways. The first one is using a new field at the packet header. The second one consists of using control signals between end-nodes and switches. The region identifier of the packet is sent, therefore, along the packet through control lines of the channel. The second approach seems to be more appealing for NoCs since bandwidth is not an scarce resource. Although the approach followed is not imposed by bLBDR we will assume the use of additional control lines. For the case of up to 8 overlapped regions we will assume 3 control lines in each channel.

Figure 5 shows the bLBDR logic at the north port with support for overlapping regions (for the E, W, S ports similar logic can be deduced). An 8-bit register is required per output port and a multiplexer. Whenever a packet is routed its region identifier is used to select the proper connectivity bit. Notice that the new logic is accessed mostly in parallel with LBDR comparators. In this case, the region support increments latency from 0.53ns to 0.66ns (modules have been designed, synthesized and mapped on a 90nm library).

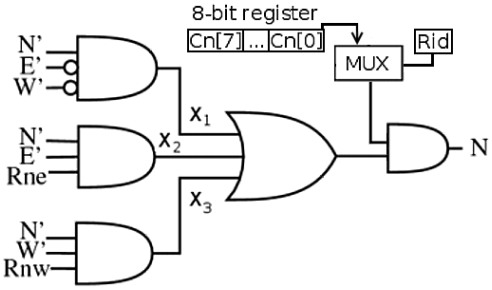

Figure 5. bLBDR with region support at the north port.

Figure 6 shows an example of valid overlapped regions in bLBDR. In this case, up to 6 regions are overlapped. R0 (region 0) is used to group 4-node sets throughout the chip (2×2 and 4×1 shapes). In total, 13 regions are defined with

R0. With R1, 3 sets of 16 nodes are built and one set of four nodes (1×4 shape). Although they do not overlap among them, each one overlaps with sets defined with R0. R3 and R4 are defined to build 8-node regions (4×2 with R3 and 2×4 with R4). A 7-node irregular region is also defined with R3. R5 splits the network into two sets, a region with the 32 upper nodes and an irregular region with the remaining 23 nodes. Finally, R2 is defined to globally access all healthy nodes in the chip (55 nodes). The set defined by this region overlaps with all the previously defined regions.

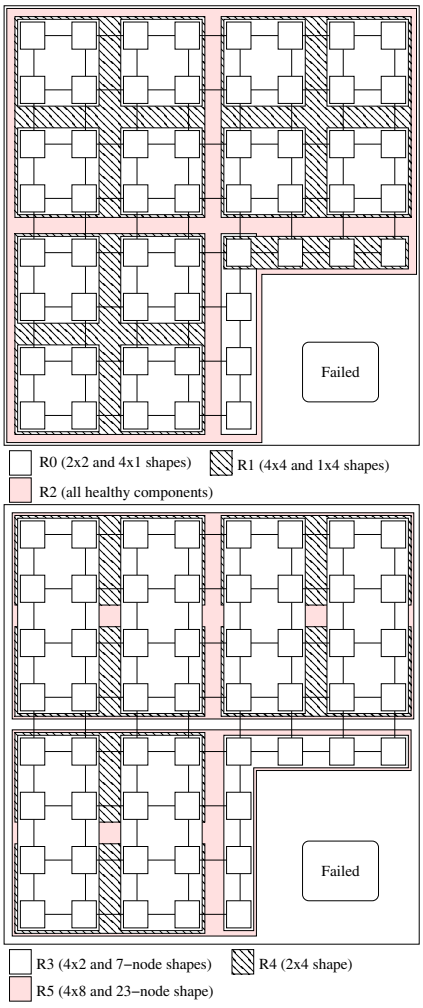

Figure 6. Example of overlapped regions.

The previous example only hints the potentials of bLBDR in defining regions. Indeed, the configuration of the connectivity bits (region definition) can be done statically or dynamically. When done statically the system can be configured with the most common regions that might be required (e.g. in a 8×8 chip, sixteen 2×2 sets, four 4×4 sets, one 8×8 set, eight 8×1 sets, ...) and depending on the application size the right region is selected. Also, different granularities of power saving are available as the system can select among several small sets of nodes (e.g. R0 sets in the example) or fewer larger sets of nodes (e.g. R3 in the example).

On the other hand, regions can be computed (and connectivity bits configured) dynamically based on application or power consumption demands. However, in this case special situations could appear during the *reconfiguration* of the connectivity bits (deadlock situations or dropped packets could appear). This is left for future research.

3.1. Deadlock Freedom and Connectivity

The mechanism presented so far can lead to deadlock or disconnected nodes if routing is not considered when partitioning the network. For example, Figure 8 shows a partitioned configuration when using the Up*/Down* (UD) routing algorithm [6]. UD is applied[1] over the entire chip not considering regions. The routing algorithm is plotted in the figure through the set of bidirectional routing restrictions it imposes. As can be deduced from the figure, nodes A and B cannot communicate through the defined region (R0 in the figure having a *d* shape). The reason is the routing restriction located at switch X. If regions were not considered both nodes would communicate through switch Y.

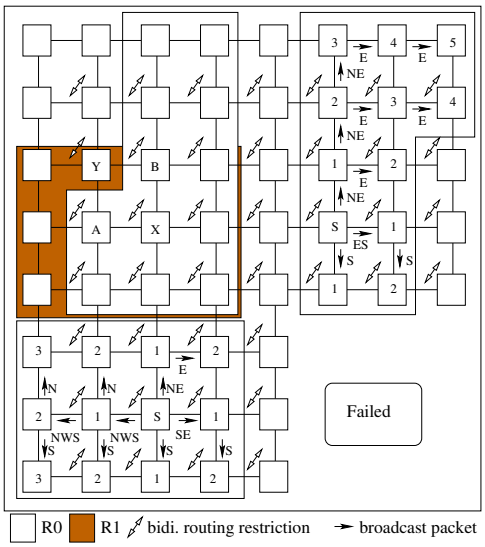

Figure 8. Regions with UD algorithm and broadcast.

One possible solution to overcome this problem would be to remove the bidirectional routing restriction at switch X. Indeed, packets belonging to R0 would not introduce any deadlock situation (the resulting channel dependency graph considering only resources in R0 would be acyclic). However, if we consider switches X and Y in an overlapped region, removing the routing restrictions may lead to deadlock. The

1. Notice that the topology and most regions are irregular, thus DOR cannot be used.

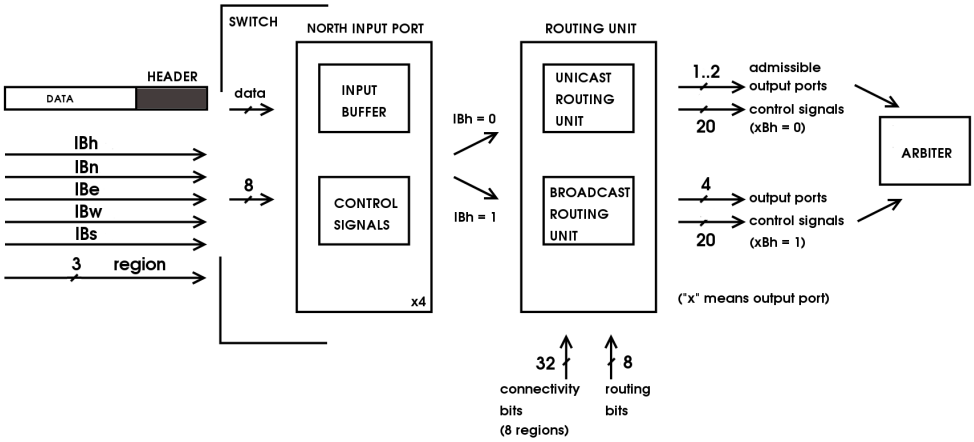

Figure 7. Detail of the router on a node with bLBDR mechanism.

cycle would be formed by packets belonging to different regions. In the example provided, a cycle could form between switches A-X-B-Y-A with packets from R0 and packets from R1.

Alternatively, using different isolated routing algorithms at each region could be thought as a viable solution to the deadlock problem. Although in isolated regions deadlock freedom is guaranteed by the proper use of the routing algorithm, using two different routing algorithms on an overlapped region may lead to deadlock. Indeed, this is similar to the deadlock problem found in dynamic network reconfiguration where the existence of two different deadlock-free routing algorithms can introduce new transient channel dependencies, thus introducing deadlock.

A simple solution to this problem, without requiring additional resources, is the selection of the routing algorithm and the shapes of the regions accordingly (the routing algorithm can be fixed first and then the shapes of the regions are defined, or the other way). As an example, the UD routing algorithm defined in the previous example can be fixed and then regions defined accordingly. Figure 8 shows the right-hand region (R0 with a *p* shape) which is equivalent to the previous one, but being rotated *180* degrees. This region is compatible with the applied routing algorithm. Notice also that square or rectangular regions are compatible with any routing algorithm.

4. bLBDR: Broadcast and Multicast

The second contribution of bLBDR is the support for multicast and broadcast operations. This is provided by benefiting from the definition of regions (either isolated or overlapped). bLBDR offers a tree-based broadcast operation inside a region. This can be viewed as a multicast operation at the chip level. The following broadcast properties are enforced by bLBDR:

- Traffic derived from a broadcast action is bounded to the region where it was initiated, even if parts of the region overlap with other regions.
- A broadcast can be initiated from any node within the region.
- Traffic originated from a broadcast action will take always minimal paths and switches (and end-nodes) will receive only one packet per broadcast action. Thus, traffic is minimized.
- bLBDR can be applied to any routing/topology combination where LBDR is used. Also, any region pattern compatible with LBDR can be used.
- The broadcast/multicast mechanism does not require any new information at switches and end-nodes. Only a small logic is required to create the broadcast tree at each switch.

For the sake of explanation, broadcast in bLBDR is described in two parts. First, when the broadcast is initiated, and second, when a switch receives a broadcast action.

4.1. A Broadcast is Initiated

When an end-node initiates a broadcast a packet is created. A control signal (B_h) is used to differentiate broadcast packets from unicast ones. Additionally, four control signals (B_n, B_e, B_w, and B_s) are used. These signals indicate the switch the directions the broadcast packet must take at the given switch. These signals travel together with the broadcast packet through control lines. As the first switch has to inject the broadcast packet through all possible directions (regardless of the connectivity), the end-node sets all the signals to one.

4.2. A Switch Receives a Broadcast Packet

A switch may receive a broadcast packet either from its local port (from the end-node) or through the incoming ports (see Figure 7). Upon reception of a broadcast packet, the switch injects through its output ports up to 4 packets (NB,

EB, WB, and SB), each one being sent through a different output port (N, E, W, S) and each one being a broadcast packet with its corresponding signals B_h, B_n, B_e, B_w, and B_s. These signals are, however, computed by the switch in a different manner. For the sake of presentation signals for packet generated for the N port are labeled NB_h, NB_n, NB_e, NB_w, and NB_s; signals for the packet for the E port are labeled EB_h, EB_n, EB_e, EB_w, and EB_s; the same for signals for packets sent through W and S ports. Additionally, broadcast signals for the incoming broadcast packet are labeled IB_h, IB_n, IB_e, IB_w, and IB_s.

Figure 7 illustrates a simplified input port (the north port) of a switch that integrates the broadcast mechanism. The input port stores both the packet and control signals. In particular, 8 control signals are received and stored: three signals encoding the region, and five signals with the broadcast info.

At the routing unit, either the unicast logic (if IB_h signal is reset) or the broadcast logic (if IB_h signal is set) is used. The unicast routing unit corresponds to the basic LBDR mechanism with the region support. This unit uses the connectivity bits and routing bits to compute the set of admissible output ports.

The broadcast unit, however, is new. This unit computes all the output ports that must be used to broadcast the packet. To do this, this unit computes all the broadcast signals (20 signals, five signals per output port). Figure 9 shows the logic required for computing each signal for each packet. The logic is replicated at each input port.

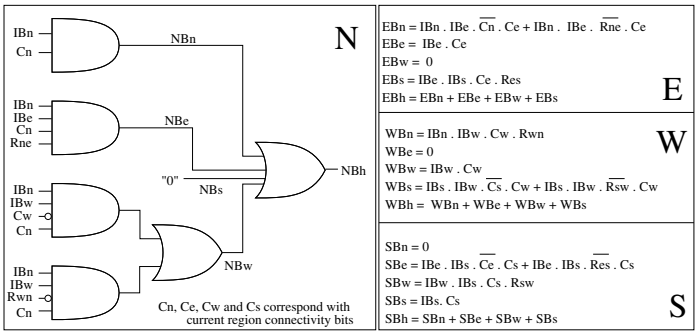

Figure 9. bLBDR logic.

Let us focus on the computed signals for the NB packet (the control signals used for the broadcast packet that will be sent through the north port; see Figure 9):

- The NB_n signal is set if the incoming broadcast packet (IB packet) has its IB_n signal set, that is, broadcast must go north and there is a connected switch (belonging to the same region) through the north port (C_n bit is set).
- The NB_e signal is set if the incoming packet must be broadcasted through the NE sector (signals IB_n and IB_e are set), there is a connected switch through the north port (C_n bit is set) and there is no routing restriction at the next switch that forbids turning towards E (bit R_{ne} is set). Notice that if the NB_e signal is set, at the same switch the EB_n signal should not be set (in order not to duplicate broadcast packets). Notice that logic for EB_n is only set if there is no connectivity through the north port (bit C_n is reset) or routing towards E is not allowed through the north port (bit R_{ne} is reset). Therefore, both signals NB_e and EB_n will not be set at the same time for the same broadcast packet.
- The NB_w signal is set if the incoming packet must be broadcasted through the NW sector (signals IB_n and IB_w are set), there is a connected switch through the north port (C_n bit is set), and either there is no switch through the W port (bit C_w is reset) or there is a routing restriction that forbids turning north at the switch reached through the W output port (bit R_{wn} is reset). Notice that in this case, signals NB_w and WB_n must not be set at the same time.
- The NB_s signal is reset since the packet will be sent through the N port.
- The NB_h signal is computed by ORing all the previous signals. If NB_h is reset, then no broadcast packet is issued through the N port.

Similar equations are obtained for the remaining output ports of the switch.

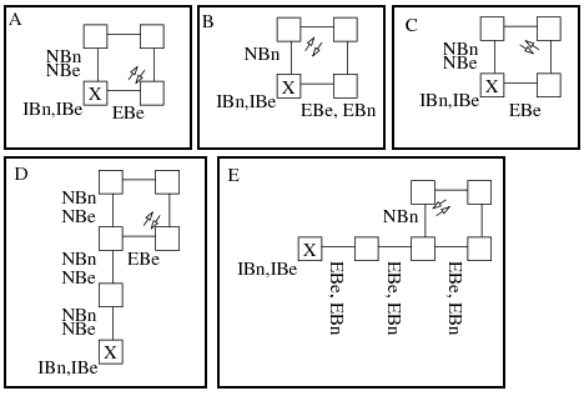

Figure 10. Different cases for the NE sector.

Figure 10 shows different situations we might face when broadcasting through the NE sector at a given switch (marked with an X). In case A, the switch forwards two broadcast packets through the N and E ports. In the NB packet the NB_n and NB_e signals are set, and in the EB packet just the EB_e signal is set. Notice that, in this situation, the E output port is used only to broadcast through the row of switches, whereas at the next switch through the N port the new NE sector is broadcasted. In this case, notice that EB_n has been reset as there is a routing restriction through the E port (bit R_{en} is reset at switch X). Contrary to this, in case B the NB_e signal is reset since there is a routing restriction through the N port (bit R_{ne} is reset). In this situation, the

N port is used only to broadcast the column of switches and the resulting new NE sector is broadcasted through the E output port.

An interesting case is C. In this situation notice that the switch has no routing restrictions through N and E ports. However, in order to avoid duplicates, the N output port is given priority. In this case, N port is used to broadcast the resulting new NE sector and E port is only used to broadcast through the row of switches [2].

In cases D and E, boundaries of the region are handled by bLBDR. In particular, in case D, the NB_n and NB_e signals are set as there is no routing restriction for the N port (bit R_{ne} is set). Notice that at the next switch the same happens and thus, the broadcast packet with its B_n and B_e signals set makes forward progress, until it reaches the switch with three neighboring switches. At that switch, case A is applied. In case E, the EB_e and EB_n signals are set. In particular EB_n signal is set because there is no connectivity through the N port (bit C_n is reset). The broadcast packet will, thus, make forward progress until it reaches the switch with three neighbors. At that switch case B is applied. The same deductions can be obtained to the remaining broadcast signals that are computed and the remaining sectors (SE, SW, and NW).

Figure 8 shows two examples of broadcast actions. Two regions are defined and within each region a node (labeled *S*) starts a broadcast action within the region. Each broadcast packet is shown in the figure as a black arrow. The letters next to the arrow indicate the broadcast control signals propagated with the packet. Inside each switch a number indicates the order in which switches receive the broadcast packet. For the rectangular defined region, three steps are required to broadcast the packet to the entire region. In a first step the *initiator* sends four copies of the packet to its neighbors. The switch at the left receives three control signals activated (*NWS*). The *S* signal is received because of the bidirectional routing restriction located at the switch below the *initiator*. The switch further broadcasts the packet through three output ports, each with different control signals, depending on the routing bits. For instance, the packet broadcasted through the *S* port only activates the *S* signal. The *W* signal is not activated, since *S* signal will be activated for the packet broadcasted through the *W* port. For the irregular region 5 steps are needed to broadcast the packet to the entire region.

4.3. Deadlock Freedom and Connectivity

In LBDR (and bLBDR) the routing algorithm is represented as a set of routing restrictions, encoded in the routing bits. The unicast routing algorithm is deadlock-free is no unicast packet crosses a forbidden routing restriction. In bLBDR packets follow unicast paths that are already deadlock-free. Indeed, notice that in the previous examples no switch forces a broadcast packet to cross a routing restriction. This is guaranteed by the fact that the broadcast logic takes into account the routing restrictions through the routing bits. As the broadcast logic enforces packets to be moved through unicast paths no deadlock can be formed, thus bLBDR is deadlock-free.

Overlapped regions do not introduce deadlock. As described in Section 3, an 8-bit register (C_x) with $C_{x[0]} \dots C_{x[7]}$ bits is used to configure the connectivity on each output port, each bit indicating the connectivity for a particular region for a given output port. When overlapped regions are allowed packets for a given region can be sent through an output port of a switch and packets belonging to a different region may be prevented (the output port has connectivity for a region but not for the other). The routing bits, however, are shared by the overlapped regions, thus all the packets (regardless the region they are using) use the same routing rules for being forwarded. Thus, none of the routing restrictions defined by the routing bits are used, therefore deadlock is prevented.

It is well known, also, that although broadcast operations are deadlock-free, concurrent broadcasts can lead to deadlock, if wormhole switching is used. This is due to the output dependencies between different packets of two broadcast actions. It has to be noted that bLBDR does not address the issue, as this is implementation-dependent and not related to bLBDR. But several solutions exist to this problem. One solution is to schedule packets at the flit-level as proposed in [11]. Additionally, using virtual cut-through switching removes the deadlock problem.

bLBDR also guarantees connectivity within a region. Any node within a region is reached by the broadcast message. For instance, imagine that an end-node initiates a broadcast action and a switch (labeled dst) is located in the same coordinate of the initiator along the X dimension, for instance along the $X-$ direction. The end-node will activate all the broadcast control signals, thus IBn will be set at the first switch indicating the packet must be broadcasted north. At the given switch the new NBn signal will be set as there is connectivity through the region (see equations in Figure 9). The same will happen in any switch located along $X-$ direction thus reaching switch dst. If the switch is located along $X+$, $Y+$, or $Y-$ directions similar deductions can be obtained (signals SBs, EBe, or WBw will be set at each visited switch along the dimension).

Now, let us consider switch dst is located in the NE quadrant (with respect to the initiator). At the first switch, the initiation will activate both signals IBn and IBe. From the equations found in Figure 9 it can be seen that either the NE sector will be broadcasted through the N port (signals

2. bLBDR provides priority to all the output ports depending on the sector that is being broadcasted. N port has higher priority than E port (for the NE sector), E port has higher priority than S port (for the SE sector), S port has higher priority than W port (for the SW sector), and W port has higher priority than N port (for the NW sector).

NBn and NBe are activated) or the E port (signals EBe and EBn are activated). Switch dst is included in the new NE sector that will be broadcasted by the next switch. Therefore, the switch will be finally reached. For the remaining sectors (SE, SW, and NW) similar deductions can be reached. Notice also that overlapped regions do not impede traffic from a region to advance through its region, therefore not disconnecting the region, and thus keeping connectedness.

5. Evaluation and Results

In this Section we evaluate bLBDR, both the broadcast and region facilities. Our goal is to evaluate the latency (in cycles) of broadcast actions when applied to different region configurations. bLBDR is compared with an equivalent unicast-based broadcast method with no region definitions. For the equivalent unicast method (named UCEQ from now on) each node starting a broadcast action sends a unicast packet to every node. The goal of the evaluation is to check whether bLBDR gets better results on latency and that initiating broadcasts on a region does not affect other regions (benefits of traffic isolation).

The scenario for the evaluations is described next. We have used noxim simulator [20], a cycle accurate network-on-chip simulator based on System C. In all simulations wormhole switching is assumed and packets are 64-flit long. Flit size is set to one byte. We have used the UD routing algorithm over an 8×8 mesh with 1-region and 4-region configurations. In the first scenario the entire NoC is used with a unique region definition. In this situation, a message is broadcasted by node 0 to all the nodes through the 8×8 topology/region. No unicast traffic is injected into the network, thus no contention is experienced by the broadcast message. In the second scenario, an irregular region (48-node irregular p topology/region) is used assuming the remaining part of the chip is powered off or disabled due to a manufacturing defect. In this scenario the same broadcast is issued and no unicast traffic is injected. Thus, both scenarios are used to test the broadcast mechanism through the entire chip.

In the third scenario, however, four broadcast messages are injected at the same time by different end-nodes (numbered, 9, 30, 36 and 59) in the 8×8 mesh with a unique region defined, thus broadcast messages will collide. In the fourth scenario (see Figure 11), the same four broadcast messages are issued at the same time but now into different isolated regions. These two last scenarios are also evaluated with an underlying traffic with a random distribution of message destinations. In the case of four regions each node injects uniform traffic to the nodes belonging to its region.

In all the cases broadcast latency is measured from the time the broadcast is initiated to the time the last end-node received the message header.

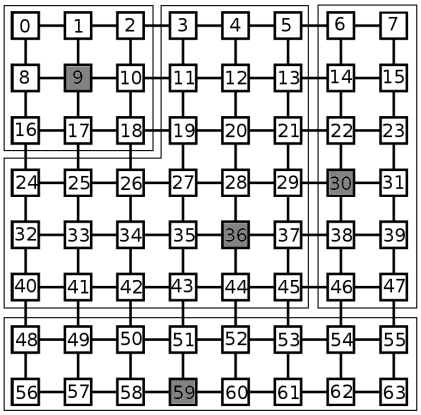

Figure 11. 4-regions scenario, one broadcast per region.

Figures 12 and 13 show the latency exhibited by the broadcast actions in all the scenarios. Figure 12 shows the cases where a unique region is used (either the 8×8 mesh or the 48-node p topology). No unicast traffic is injected. As can be noticed, a tree-based broadcast mechanism (bLBDR) minimizes the latency. For the 8×8 mesh, only 94 cycles are required for bLBDR where as 4062 cycles are needed for UCEQ, a reduction factor of 129. Results for the p topology are similar (86 cycles for bLBDR and 3030 cycles for UCEQ). This is an obvious result obtained from the fact that using a tree-based broadcast (bLBDR) traffic is minimized and most of the messages are generated in parallel at the branches of the broadcast tree.

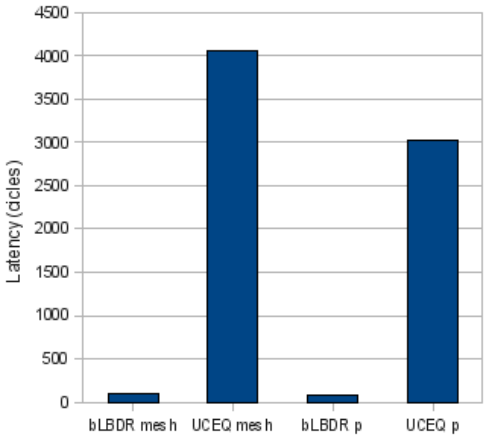

Figure 12. Latency on a unique region.

Figure 13 shows more interesting cases where underlying unicast traffic is assumed and/or regions are defined. The first two columns shows the effect of using isolated regions to bound the multicast traffic. In both cases four broadcast packets are triggered, however in the first case each multicast is confined to an isolated region (see Figure 11), where as in the second case regions are not defined, thus the four broadcasts collide in the network. When using regions all

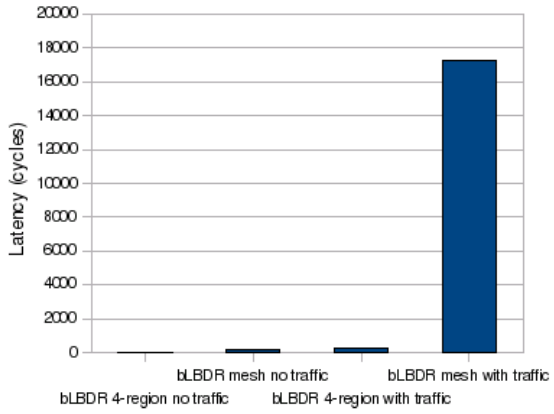

Figure 13. bLBDR latency with different scenarios.

multicast operations end up in 77 cycles, where as 275 cycles are needed when using no region definitions (reduction factor of 3.5).

Finally, the last two columns in Figure 13 show the same case when random unicast traffic is injected through the entire network. In this case, when using bLBDR with regions only 303 cycles are needed, where as 17337 cycles are needed with no region definitions (reduction factor increased to 57). Again, the savings in latency come by the fact of isolating the traffic and eliminating the network contention between different broadcast messages.

5.1. Area, Delay, and Power Evaluation

In this Section we evaluate the area, power, and delay of an implementation of bLBDR. Also we compare the results with other routing mechanisms. In particular, several routing modules have been implemented: LBDR, LBDR with regions (*reg* in the table), bLBDR (LBDR + regions + broadcast), FSM-based module implementing DOR algorithm (XY), table-based routing (RT) and region-based[3] (RBR) routing.

When using routing tables (RT) a table is required at the switch with as many entries as the number of nodes and input ports. Even more, every entry needs to store different output ports returned by the routing function. Hence the cost of this alternative is $N \times d \times d$, where N is the number of nodes and d is the number of ports. However, if we consider mesh topologies and minimal routing, the set of admissible output ports returned by the routing function has cardinality of two at maximum. Therefore, in this case, the routing table has a cost of $N \times 2 \times d$ bits. This overall requirement, for memory-based solutions, grows linearly with network size. On the other hand, the minimum logic to support unicast routing with LBDR requires no more than two comparators, five gates, two routing bits, and one connectivity bit per output port. For LBDR with regions (*reg*) an 8-bit register and a multiplexer is needed in addition per output port. For the full deployed bLBDR mechanism (broadcast and multicast support within regions) 24 more gates are added per output port to the routing logic.

Region-based routing (RBR) is a mechanism to support efficient routing algorithms in NoCs, described in [13], and requires 30 gates, four registers of size $log_2(\mathrm{N})/2$, one register with *d+1* bits and one register with d bits per region. In the analysis 16 regions are used.

All the modules have been designed and synthesized using Synopsys Design Compiler and mapped on a 90nm technology library from TSMC. An 8×8 NoC mesh has been considered for the evaluation. Area, power, and delay figures of all module-implemented switches can be seen in Table 1.

As expected, LBDR, LBDR with regions (*reg*) and bLBDR are less expensive in terms of area, power and delay than routing tables (RT). For instance, bLBDR, requires approximately 13% of the original area required for RT. In fact, both logic units, LBDR and bLBDR, are about 18% of the original area consumed by RT and require 17x less power than RT. In terms of delay, factors are not so much greater. Compared to FSM-based switches, LBDR and bLBDR are more expensive in terms of area and delay. For instance, LBDR has a requirement of 2x area and bLBDR has a factor of 5.5x. But both have similar numbers in terms of power, LBDR being cheaper than XY and bLBDR being similar. This addition in area and latency, however, is balanced by the benefits of enabling virtualization and broadcast support, not achievable with XY (DOR) routing.

Table 1 also shows the breakdown of area, power, and delay in the switch. bLBDR module affects the overall area and power budget by only a few percent. As can be seen in the table, the contribution in both silicon area and power dissipation of the block implementing the bLBDR mechanism is negligible as compared with the FIFO buffers and crossbar. FIFO buffers and crossbar dominate the design both in terms of area and power consumption.

6. Related Work

When dealing with irregular topologies, routing can be implemented as source routing or distributed routing. In source routing, the source end-node computes the path and stores it in the packet header. Since the header itself must be transmitted through the network, it consumes network bandwidth. The Teraflop research chip from Intel uses source routing. In distributed routing, however, each switch computes the next link that will be used while the packet travels across the network. The packet header only contains the destination ID. One interesting property in distributed routing is that adaptivity can be achieved by providing alternative routing options at each hop.

3. Region-based routing has been proposed as a way to pack the routing info found in a routing table, and by no means region-based routing isolates traffic into regions.

Table 1. Area, power and delay evaluations. 8×8 mesh network.

	LBDR	reg	bLBDR	XY	RBR	RT	Arbiter	FIFO	XBar
Area (μm^2)	684.432	1728.720	2049.768	366.912	1772.467	14986.943	910.224	40286.230	6648.164
Delay (ns)	0.530	0.660	0.660	0.430	0.470	0.740	0.140	0.570	1.240
Power (μW)	104.105	209.403	235.557	202.947	391.975	4067.303	364.511	12348.972	4168.241

Distributed routing can be implemented in different ways. The approach followed in regular topologies is the so called algorithmic routing, which relies on a combinational logic circuit that computes the output port to be used as a function of the current and destination nodes and the status of the output ports. The implementation is very efficient in terms of both area and speed, but the algorithm is specific to the topology and to the routing strategy used on that topology. To deal with non-regular topologies, switches based on forwarding tables were proposed. In this case, there is a table at each switch that stores, for each destination end-node, the output port that must be used. This scheme can be easily extended to support adaptive routing by storing several outputs in each table entry. The main advantage of table-based routing is that any topology and any routing algorithm can be used, including fault-tolerant routing algorithms. However, memories, do not scale in terms of latency, power consumption, and area, thus being impractical for NoCs.

Possibly, the size of the routing table can be reduced in some environments. This is the case of application-specific systems [14], [15] where the communication pattern may be known in advance. However, is not the case for generic purpose multi-core chips.

The concept of virtualization is not a new one. It has been applied to different domains for different purposes: virtual machines, virtual memory, storage virtualization, and virtual servers in data centers. In multi-core, recent proposals have been made to provide virtualization. One first example is the proposal of virtualization at the IP level [7]. In this work the authors conceptually provide mechanisms to assign IP resources to different applications. One second example is proposed in [10] where authors propose the virtualization at the memory level, providing means to distribute memory resources to different applications. No insights are provided, however, at the on-chip network level in none of these two contributions. A virtualized NoC may be viewed as a network that partitions itself into different regions, each region serving different applications and traffic flows. bLBDR tries to complement both virtualization efforts, at the IP level, memory level, and NoC level.

Basically, there are two ways to provide multicast (or broadcast). One is path-based where the packet is sent from the source to the first destination, from that destination to the second destination, and so on. In this kind of multicast, the latency of the multicast operation is high, as packets are sent sequentially. In the second type of multicast a tree is formed to reach all the destinations (tree-based multicast). At each switch decisions are made to initiate branches of the tree. The Virtual Circuit Tree Multicasting (VCTM) mechanism [11] follows this trend. One of the problems of tree-based multicast routing is the resource requirements at each switch since the tree is created and removed dynamically.

The work presented in [11] is very solid and proposes a very elegant and efficient mechanism to support multicast and broadcast communication. However, the main differences with the bLBDR mechanism are threefold. First, VCTM relies on the use of dimension order routing (DOR) to forward packets (either unicast or multicast). DOR is not suitable for irregular topologies (induced by manufacturing defects or due to application mapping algorithms). Second, VCTM does not provide a support for virtualizing the multi-core chip. Indeed, VCTM multicast support is constrained to the case when the entire chip can be used by a single application. Third, memory resources are required either at the end-nodes and at the switches to support VCTM. In particular, the most demanding resource comes at the switch where multicast tables are required. bLBDR, on the contrary, requires much less logic since it only requires an 8-bit register per output port and a small set of logic gates.

There are few works proposing the use of multicast routers for NoCs. Some of them [16], [17] rely on building virtual circuits, thus being latency sensitive at the circuit establishment. For off-chip multicast routers, please refer to [18], [19] (for multistage networks and/or high-radix switches). These techniques are not suitable for on-chip networks.

One important issue of the previous multicast mechanism is that they rely on the dimension order routing (DOR) to guarantee deadlock. As multicast packets are sent through DOR routes, they do not introduce any deadlock. This is a limiting factor in these proposals since it impedes them to be applied on other topologies or environments where DOR can not be applied. A simple case is an irregular topology (induced by a manufacturing defect or due to a mapping result of a new application in the system) that has an L shape. bLBDR, however, embeds in the same routing mechanism the ability to implement topology-agnostic routing algorithms with unicast and multicast support. This is a key property of bLBDR not present in previous proposals.

7. Conclusions

In a multi-core processing chip, a network-on-chip is required to interconnect the cores. These networks need to be carefully designed to meet the constraints of power

consumption, area, and ultra low latencies. Although there are viable solutions, like DOR and the use of 2D meshes, new challenges need to be addressed that are not fulfilled with existing solutions.

To the best of our knowledge, this is the first time that a routing mechanism for NoCs fulfills all the practical requirements for this kind of networks, including not only low latency and power and area efficiency, but also support for virtualization, partitionability, fault-tolerance, traffic isolation, and broadcast across the entire network as well as constrained to coherency domains or regions. This support is very convenient if not mandatory to tolerate manufacturing failures, coherency protocols, and so on.

In particular, in this paper we have proposed an extension of the LBDR routing mechanism to support the definition of isolated and/or overlapped domains within the network, and the use of multicast and/or broadcast in an efficient manner. bLBDR is able to define multiple region configurations with a very small logic block. Models mapped on a 90nm technology library have demonstrated that area, delay, and power requirements are much less than the ones required by routing tables, either at switches or end-nodes.

The applicability of bLBDR and LBDR for chip/system virtualization is meant to be deeply analyzed. bLBDR and LBDR allow NoC partitioning (we want to assure coherency and isolation for every part, e.g. different running applications) or dealing with failures.

The bLBDR mechanism proposed in this paper does not follow any of the typical approaches (path-based or tree-based multicast). Indeed, bLBDR can be viewed as a region-based multicast routing mechanism, where regions are predefined based on the current set of applications being executed on the chip. The mechanism, however, can be adapted to support dynamic formation of regions by changing the connectivity bits, and thus the shape of the regions. This, however, is left for future research.

Acknowledgments

The authors would like to thank Dr. Maurizio Palesi for his help on providing power and area results. 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 Grants PBC-05-005-2 and PCC08-0078.

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, Num. 4/5, 2005.

[3] K. Sankarakingam, R. Nagarajan, H. Liu, C. Kim, J. Huh, D. Burger, S. Keckler, and C. Moore, "Exploiting ILP, TLP, and DLP using Polymorphism in the TRIPS Architecture," in *Int. Symp. on Computer Architecture*, 2003.

[4] M.B. Taylor, W. Lee, S.P. Amarasinghe, and A. Agarwal, "Scalar Operand Networks: On-Chip Interconnect for ILP in Partitioned Architecture," in *Int Symp. on High-Performance Computer Architecture*, 2003.

[5] S. Swanson, K. Michelson, A. Schwerin, and M. Oskin, "Wavescalar," in *Int. Symp. on Microarchitecture*, 2003.

[6] M.D. Schroeder, A.D. Birrell, M. Burrows, H. Murray, R.M. Needham, T.L. Rodeheffer, E.H. Satterthwaite, and C. Thacker, "Autonet: A High-speed, Self-configuring Local Area Network using Point-to-point Links," Technical Report SRC research report 59, DEC, April 1990.

[7] N. Aggarwal, P. Ranganathan, N. P. Jouppi, and J. E. Smith, "Configurable Isolation: Building High Availability Systems with Commodity Multi-Core Processors," in *Int. Symp. on Computer Architecture*, 2007.

[8] J. Laudon and D. Lenoski, "The SGI Origin: a ccNUMA Highly Scalable Server," in *Int. Symp. on Computer Architecture*, 1997.

[9] M.M.K. Martin, M.D. Hill, and D.A. Wood, "Token Coherence: Decoupling Performance and Correctness," in *Int. Symp. on Computer Architecture*, 2003.

[10] M. Marty and M. Hill, "Virtual Hierarchies to Support Server Consolidation," in *Int. Symp. on Computer Architecture*, 2007.

[11] N.E. Jerger, L.-S. Peh, and M. Lipasti, "Virtual Circuit Tree Multicasting: A Case for On-Chip Hardware Multicast Support," in *Int. Symp. on Computer Architecture*, 2008.

[12] J. Flich, S. Rodrigo, and J. Duato, "An Efficient Implementation of Distributed Routing Algorithms for NoCs," in *Int. Symp. on Networks on Chip*, 2008.

[13] J. Flich, A. Mejía, P. López and J. Duato, "Region-Based Routing: An Efficient Routing Mechanism to Tackle Unreliable Hardware in Network on Chips," in *Int. Symp. on Networks on Chip*, 2007.

[14] M. Palesi, S. Kumar, R. Holsmark, "A Method for Router Table Compression for Application Specific Routing in Mesh Topology NoC Architecture," in *SAMOS conference*, 2006.

[15] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "Routing Table Minimization for Irregular Mesh NoCs," in *Design, Automation and Test in Europe*, 2007.

[16] J. Liu, L.-R. Zeng, and J. Tenhunen, "Interconnect Intellectual Property for Network on Chip," in *Journal of System Architectures*, 2004.

[17] Z. Lu, B. Yin, and A. Jantsch, "Connection Oriented Multicasting in Wormhole-switched Networks on Chip," in *Emerging VLSI Technologies and Architectures*, 2006.

[18] C.B. Stunkel, J. Herring, B. Abali, and R. Sivaram, "A New Switch Chip for IBM RS/6000 SP Systems," in *Supercomputing*, 1999.

[19] J. Turner, "An Optimal Nonblocking Multicast Virtual Circuit Switch," in *Infocom conference*, 1994.

[20] Noxim: Network-on-Chip simulator, available at http://noxim.sourceforge.net.