An Efficient Implementation of Distributed Routing Algorithms for NoCs *

J. Flich, S. Rodrigo, and J. Duato
Parallel Architectures Group
Technical University of Valencia, Spain
email {jflich, srodrigo, jduato}@disca.upv.es

Abstract

The design of NoCs for multi-core chips introduces new design constraints like power consumption, area, and ultra low latencies. Although 2D meshes are preferred, heterogeneous blocks, fabrication faults, reliability issues, and chip virtualization may lead to the need of irregular topologies or regions. In this situation, efficient routing becomes a challenge. Although the use of routing tables at switches is flexible, it does not scale in terms of latency and area due to its memory requirements.

LBDR (Logic-Based Distributed Routing) is proposed as a new routing method that removes the need of using routing tables at all. LBDR enables the implementation of many routing algorithms on most of the practical topologies we might find in the near future in a multi-core system. From an initial topology and routing algorithm, a set of three bits per switch/output port is computed. Evaluation results show that, by using a small logic, LBDR mimics the performance of routing algorithms when implemented with routing tables, both in regular and irregular topologies.

1 Introduction

Multi-core architectures are becoming mainstream for designing high performance processors. As power limits the performance for single-core solutions, designers are shifting to the multi-core domain where 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. As an example, the TeraScale chip has been recently announced with 80 cores [1].

Such a large number of cores requires a high-performance on-chip interconnect (NoC) to efficiently communicate cores among them and with cache blocks and/or memory controllers. Current chip implementations use simple network structures such as buses or rings [2]. However, as the number of cores increases such networks become the bottleneck of the system, thus bus and ring topologies become unpractical. 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 the case of the TeraScale chip.

For routing purposes, logic-based routing (e.g. Dimmension Order Routing; DOR) is preferred to reduce latency, power, and area requirements for the NoC. The high integration scale, however, pushes a number of communication reliability issues. Crosstalk, power supply noise, electromagnetic and inter-symbol interference are some of these issues. Moreover, fabrication faults may appear, in the form of defective core nodes, wires or switches. In these cases, while some regions of the chip are defective, the remaining chip area may be fully functional. From the NoC point of view, in presence of such fabrication defects, the initial regular topology has become an irregular one. This is the case for topologies shown in Figure 1.

Additionally, in order to exploit the increasing number of cores, and due to the fact that applications are not getting enough parallelism, virtualization of the chip is becoming a necessity. In a virtualized system resources are distributed among different tasks or applications. The network must guarantee traffic isolation within regions, thus, leading to irregular sub-networks within the original 2D mesh. Figure 1.h shows an example where three regions are used.

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 TeraScale chip 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.

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

*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.

978-0-7695-3098-7/08 $25.00 © 2008 IEEE
DOI 10.1109/NOCS.2008.28

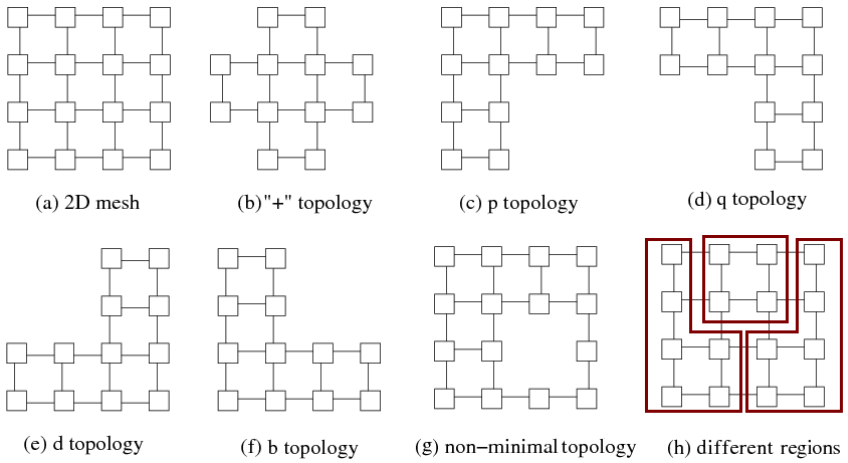


Figure 1. Examples of topologies (a-f and h) supported and (g) not supported by LBDR.

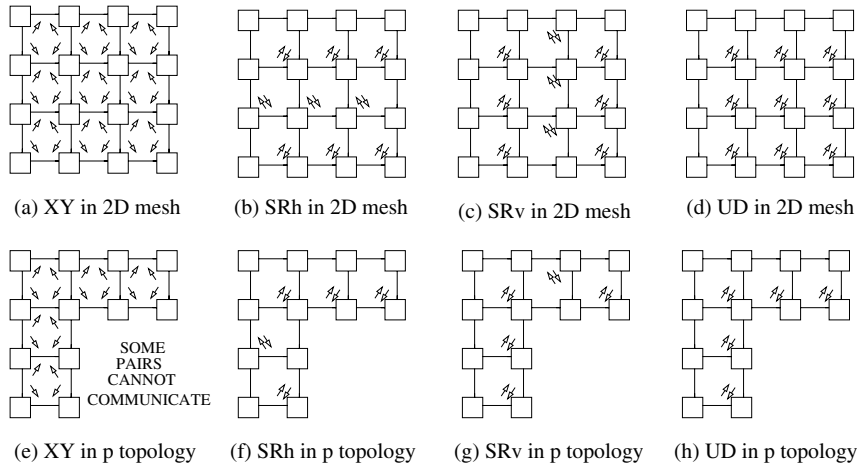


Figure 2. Examples of routing algorithms (by their routing restrictions) in 2D mesh and p topology.

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 where the communication pattern may be known in advance. However, is not the case for generic purpose multi-core chips.

It would be interesting to find an implementation, for irregular topologies (partial 2D meshes), that allows the use of any distributed routing algorithm without the need of using routing tables nor source-based routing. In this paper we take on such a challenge. We propose a very simple mechanism that removes the routing tables at every switch, thus enabling the distributed implementation of any routing algorithm on irregular topologies. The mechanism, referred to as logic-based distributed routing (LBDR), relies on three bits per output port at every switch and a small logic of several gates.

The rest of the paper is organized as follows. In Section 2 related work is presented. Then, in Section 3 the system environment of LBDR is shown. In Section 4 the mechanism is presented and in Section 5 its deadlock-freedom and connectivity properties are demonstrated. In Section 6 some evaluation results are presented and the main benefits of LBDR are exposed in Section 7. The paper is concluded with Section 8.

2 Related Work

Some work on the reduction of memory requirements for routing in NoCs already exists. One solution is Interval Routing [3]. With interval routing, sets of destinations requesting the same output ports are grouped. This method is specific for regular topologies. The FIR method [4] is an extension of interval routing for allowing different routing algorithms in meshes and tori networks. However, FIR is not applicable to irregular networks. Another solution is named *street-sign routing* [12]. In this method, only the router name of the next turn and the direction of the turn are included in the packet header.

Recently, two solutions for irregular topologies have been proposed [5], [6]. In both cases, the destinations are grouped into regions and regions are coded into switches. A region is coded by the top left-most switch and the bottom right-most switch. Although the number of regions grows logarithmically with the number of failures, the number is unbounded and each region implies a logic. Another solution for routing table minimization is presented in [8]. In this case logic is used for the *regular* case and a deviation routing table is used for routing deviations.

Although different solutions exist, none of them allows the implementation of distributed routing algorithms in irregular topologies with no routing tables and minimum logic.

3 System Environment

For the sake of simplicity we focus on networks with no virtual channel requirements, and assume wormhole switching (although the proposed method also works for virtual cut-through switching as well). Messages (or packets in virtual cut-through) are routed with X and Y offsets assuming the X and Y coordinates of the final destination are included in the message header (X_{dst} and Y_{dst}), and each switch knows its X and Y coordinates (through the X_{curr} and Y_{curr} registers at each switch).

LBDR can be applied to a combination of topologies and routing algorithms with some particular characteristics. The following paragraphs describe the conditions topologies and routing algorithms must meet.

The typical topology of choice for NoCs is the 2D mesh network. However, with the advances of technology, other topologies may be suitable for NoCs. As the number of nodes increases some NoC components may fail. Therefore, topologies derived from an initial 2D mesh, but with some manufacturing defects or failures may come up. For instance, Figure 1 shows different topologies. Due to manufacturing defects some parts of the topology have been disabled for normal operation. This is the case for topology in Figure 1.c (p topology). In this case some nodes and links have been disabled. Equivalent topologies are the ones shown in Figures 1.d (q topology), 1.e (d topology), and 1.f (b topology). Obviously, there are other topologies with the same shape (p, q, d, or b) but with different number of nodes and links. Additionally, other circumstances may lead to the need for using irregular topologies. Examples are application-specific systems and chip/server virtualization (an example is shown in Figure 1.h).

It should be noted that all the described topologies share the same property: all the end-nodes (assuming at least one end-node attached to each switch) can communicate with the rest of nodes through any minimal path defined in the original mesh topology (the topology pictured in Figure 1.a). LBDR can be applied to all the topologies that fulfill this property. LBDR is, however, not applicable to topologies where some pairs of end-nodes cannot communicate through a minimal path defined in the original 2-D mesh topology. Figure 1.g shows an example.

Note that topologies with several disabled regions are suitable for LBDR. One example is shown in Figure 1.b. Notice that in this case all the end-nodes can communicate through minimal paths defined in the original 2-D mesh topology.

A deterministic (or partially adaptive) routing algorithm without cyclic dependencies among links or buffers can be represented by the set of routing restrictions it imposes. As an example, Figure 2 shows the routing restrictions defined by XY, SR_h [6], SR_v [6], and UD (up*/down*) [7] routing algorithms on a 2-D mesh topology and a p topology. Each arrow indicates a routing restriction. Basically, a routing restriction forbids any packet to use two consecutive channels. So, the final paths for each pair of communicating end-nodes will not pass through any routing restriction. In this paper we define a routing restriction as the pair of channels that can not be used in sequence by any packet. For instance, at the first (top left-most) switch in Figure 2.a there is a SE restriction [1].

As can be seen in the Figure, the XY routing algorithm is designed only for the 2D mesh topology and it is unsuitable for non-regular topologies. Topology-agnostic routing algorithms like SR_h, SR_v, and UD can, however, be applied to any topology.

LBDR is applicable to any routing algorithm that complies with the following condition: defining the following two sets of channels {N,S} and {E,W}, all the routing restrictions are formed by two channels, each one from a different set. Thus, for instance, restriction EW is not allowed. In other words, routing restrictions forbid only some changes in the direction. Notice that this makes sense since it allows for minimal routing (restrictions like WE, EW, NS, SN, SS, NN... always force the need for non-minimal paths). Notice that all the routing algorithms pictured at Figure 2 comply with this requirement, since all the restrictions are: NE, NW, SE, SW (in the case of XY), WN, NW, NE, EN (in the case of SR_h), WN, NW, WS, SW (in the case of SR_v), and WN, NW (in the case

[1] Channels are labeled as N (North), E (East), W (West), and S (South).

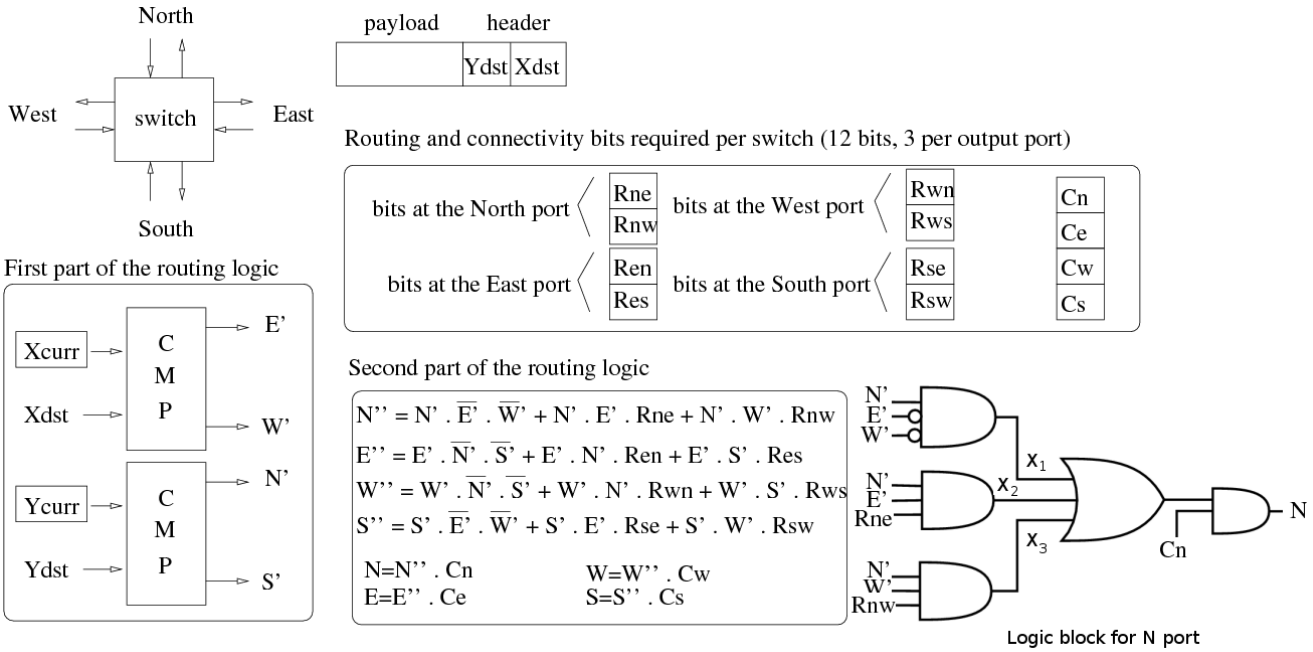


Figure 3. The LBDR method.

of UD).

Other routing algorithms like FX [9] and Turn Model [10] also adhere to these conditions, thus they can be implemented with LBDR.

4 LBDR Description

Figure 3 shows the details of LBDR. It relies on the use of only three bits per switch output port. Therefore, 12 bits in total per switch. The value of these bits depends on the topology and the routing algorithm being implemented, and are computed and uploaded to the switches before normal operation (at system boot).

Bits are grouped in two sets: routing bits and connectivity bits. Routing bits indicate which routing options can be taken, whereas connectivity bits indicate whether a switch is connected with its neighbors.

Regarding the routing bits, the bits for the E output port are labeled R_{en} and R_{es}. They indicate whether packets routed through the E output port may take the N port or S port at the next switch, respectively. In other words, these bits indicate whether packets are allowed to change direction at the next switch. For output port N the bits are accordingly labeled R_{ne} and R_{nw}, for output port W R_{wn} and R_{ws}, and for output port S R_{se} and R_{sw}.

Regarding the connectivity bits, each output port has a bit, referred to as C_x indicating whether a switch is connected through the x port. Thus, connectivity bits are C_n, C_e, C_w, and C_s.

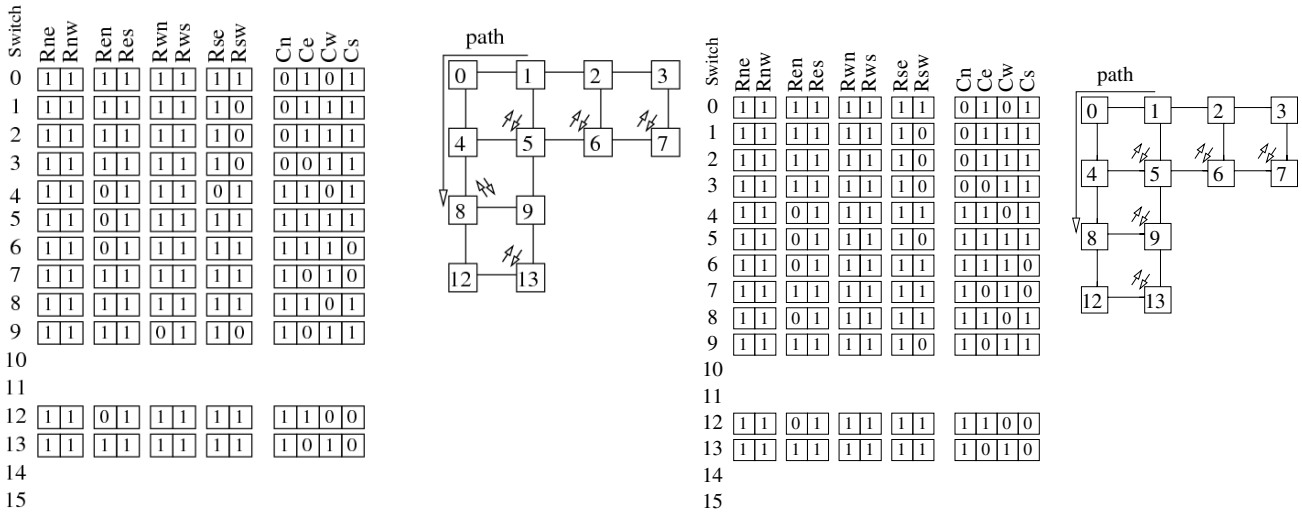
Figure 4 shows two examples of all the bits at every switch for an irregular topology when using two different routing algorithms: SR_h and UD.

Routing logic of LBDR is divided in two parts (see Figure 3). The first part computes the relative position of the packet's destination. For this, two comparators are used and X_{curr} and Y_{curr} are compared with X_{dst} and Y_{dst}. At the output of this logic one or two signals may be active (e.g. if the packet is in the NW quadrant then N' and W' signals are active). Note also that packets forwarded to the local port are excluded from the routing logic.

Once the N', E', W', and S' signals are computed, the second part of the logic comes into play. It consists of four logic units, one for each output port. Each one can be implemented with only two inverters, four AND gates and one OR gate. As all of them are similar we describe here only the logic associated with the N output port.

The N output port is considered for routing the incoming packet when either one of the following three conditions is met. If none of the conditions is met, then the N port can not be considered for routing the packet (additionally, the connectivity bit C_n is inspected in order to filter the N port):

- The packet's destination is on the same column ($N' \times \overline{E'} \times \overline{W'}$).
- The packet's destination is on the NE quadrant and the packet can take the E port at the next switch through the N port ($N' \times E' \times R_{ne}$).
- The packet's destination is on the NW quadrant and the packet can take the W port at the next switch through the N port ($N' \times W' \times R_{nw}$).



(a) SR_h routing (b) UD routing

Figure 4. Example of LBDR for an irregular (p) topology.

Notice that, for example, N and E signals could be active at the same time. In this case, the switch has to choose among them in the arbiter unit, according whether adaptiveness is allowed or the routing algorithm is deterministic.

LBDR will mimic performance in most of the routing algorithms. This is the case for the XY and UD routing algorithms. In these algorithms, the routing restrictions are located in the same relative position through all the rows and columns. As an example, Figure 4.b shows all the bits for the UD routing algorithm in a p topology. Imagine the path from source 1 to destination 8 as it is shown in Figure 4.b. Notice that output port S is not taken at switch 1 because there is a NW routing restriction at switch 5 (bit R_{sw} is zero at switch 1). This decision does not impact performance, as the S output port cannot be taken to forward properly the packet (packet would never be able to turn to W in the column). This is a very interesting observation as it allows LBDR to achieve the maximum performance of the routing algorithm with very low logic/area requirements (the same applies for XY routing in a 2D mesh topology).

However, there are situations (e.g. more advanced routing algorithms like SR) where LBDR induces some inefficiencies. An example can be seen in Figure 4.a. In this case, like before, switch 1 decides to discard output port S because its R_{sw} bit is not active (there is a NW restriction at the next switch through the S port). However, in this case, a valid path would be 1-5-9-8. Therefore, LBDR reduces adaptiveness. Although LBDR is still working (it always provides a valid set of paths) the performance degradation could be unacceptable (see Section 6). In order to fix this, we extend in the next section the mechanism to overcome this problem. However, bear in mind that if we change the routing algorithm to UD, the problem disappears.

It is important to note that only the bits referring to a routing restriction are set to zero and the remaining ones are set to one, even those that refer to switches not existing in the topology (for instance bit R_{nw} at switch 0). However, they must be set to one in order the mechanism to work properly. This can be better seen through an example. Imagine the path at Figure 4.a from switch 13 to switch 7. At switch 13 the signals N' and W' are active. Also, signals N'' and W'' are active. In particular, N'' is activated as R_{ne} at switch 13 is set to one, although it does not make sense for routing purposes. However, this allows the packet to being forwarded north, until it reaches switch 5, where it can take the east direction. Notice that output port E will never be taken at switches 13 and 9 due to the connectivity bit C_e.

Deadlock freedom is guaranteed by the underlying routing algorithm. If packets do not cross any routing restriction then no cycle can be formed. Notice that a packet is forwarded by LBDR by using the R_{xy} routing bits, thus, ensuring no routing restriction is crossed. Connectivity is also guaranteed since LBDR uses all the possible minimal paths provided by the underlying routing algorithm. Formal demonstrations can be found on Section 5.

4.1 LBDRe

Figure 5 describes the extended LBDR method. We refer to is as LBDRe. As can be seen, the routing and connectivity bits (3 bits per output port) are still maintained, and they are computed in the same way. Four new bits per switch output port are, however, added.

The bits labeled $R2_{xy}$ indicate whether the y direction can be taken two hops away from the current switch through the x direction. For example, $R2_{ne}$ indicates whether a packet is allowed to change direction to E at the switch

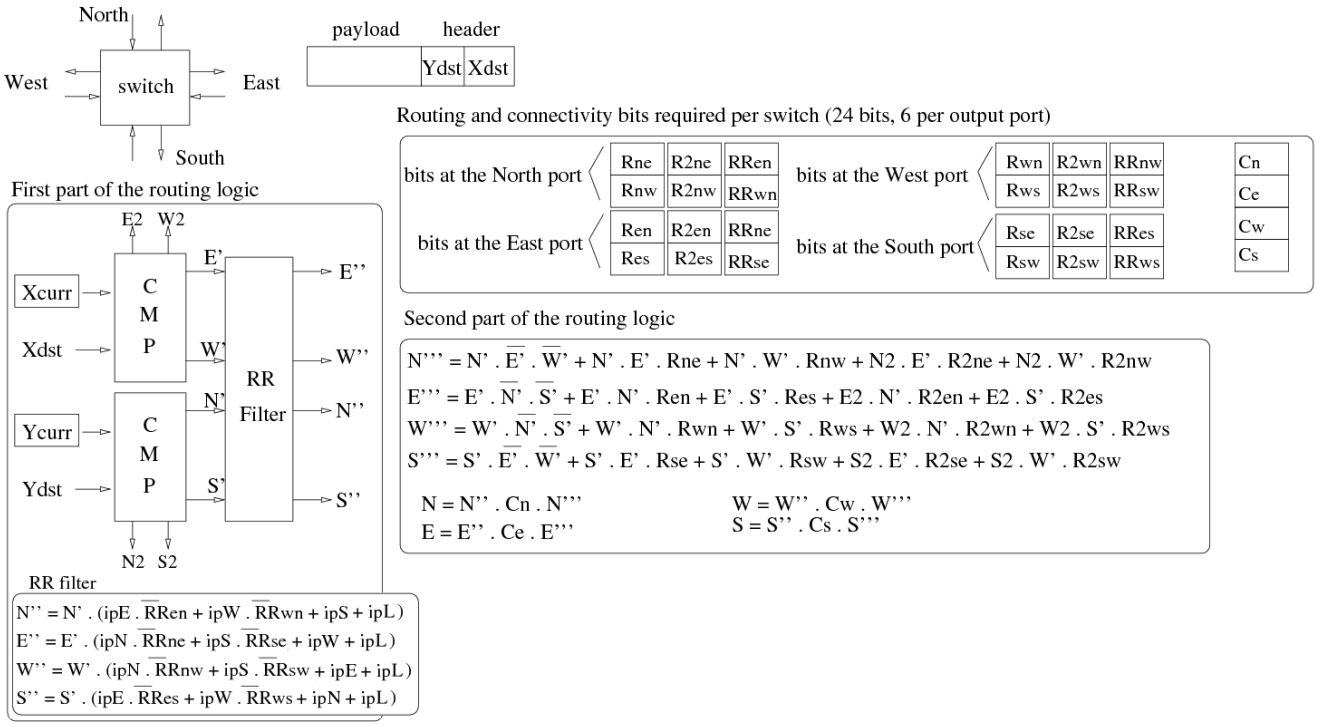


Figure 5. The LBDRe method.

located two hops in the N direction. Notice that this set of bits have similar meaning with the ones used in LBDR. In some sense, these bits provide visibility to the switch of the routing possibilities two hops away. However, it must be stated that these bits must be not active if they refer to a non-existing switch. For instance, in Figure 6 the $R2_{nw}$ bit at switch 4 is not active.

The bits labeled RR_{xy} indicate whether there is a routing restriction between x and y channels at the current switch. These bits are needed in order to avoid the formation of cycles, which is described in the example below.

To sum up, LBDRe requires 24 routing bits grouped by 6 bits per output port. Figure 6 shows an example of the LBDRe bits for a *p* topology. Additionally, the switch needs five internal signals ipN, ipE, ipW, ipS and ipL to indicate the incoming port of the packet being routed.

The first part of the routing logic is slightly augmented compared to LBDR. In particular, based on the X and Y coordinates of the current switch and the packet's destination, the logic computes the relative directions N', E', W', and S'. Additionally, four extra signals $N2$, $E2$, $W2$ and $S2$ are computed. These signals are active if the packet's destination is at least two hops away in the corresponding direction (if $N2$ is active, then at least two hops must be done in the N direction to get closer to packet's destination). Notice that these signals can be easily computed with additional comparators with the X_{curr} and Y_{curr} coordinates shifted in one position.

The first part of the logic is also in charge of inhibiting the possible output ports that would lead crossing a routing restriction. For this, the RR (routing restriction) filter logic is used. This logic requires two inverters, three AND gates and one OR gate per output port. The resulting signals are labeled as N'', E'', W'', S''. They feed the final part of the logic.

The second part evaluates the routing options at the one-hop and two-hops neighbors. For this, the previous logic functions for LBDR have been extended. For instance, for the output port N, the port will be selected if any one of the following conditions are met:

- The packet's destination is on the same column ($N' \times \overline{E'} \times \overline{W'}$).
- The packet's destination is on the NE quadrant and the packet can take the E port at the next switch through the N port ($N' \times E' \times R_{ne}$).
- The packet's destination is on the NW quadrant and the packet can take the W port at the next switch through the N port ($N' \times W' \times R_{nw}$).
- The packet's destination is on the NE quadrant, the packet's destination is at least two hops away through the N port, and the packet can take the E port at the two-hops neighbor switch through the N port ($N2 \times E' \times R2_{ne}$).

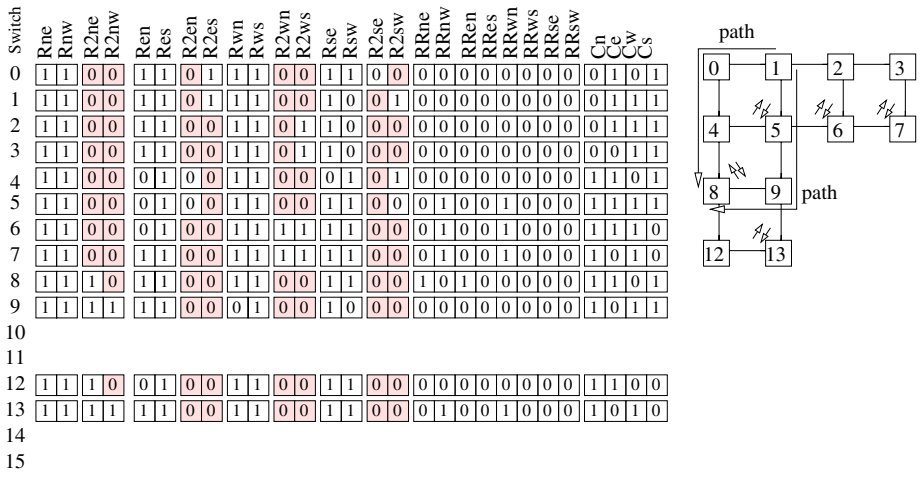


Figure 6. Example of LBDRe for an irregular (p) topology and SR_h routing algorithm.

- The packet's destination is on the NW quadrant, the packet's destination is at least two hops away through the N port, and the packet can take the W port at the two-hops neighbor switch through the N port ($N2 \times W' \times R2_{nw}$).

Finally, the connectivity bit C_n and the routing-restriction filter (N'') are used to filter the output port. For the remaining ports, similar deductions are considered.

Notice that the LBDRe mechanism solves the problem found with LBDR. Figure 6 shows two possible paths from source 1 to destination 8. At switch 1, the S output port can now be taken because the $R2_{sw}$ bit is active and the internal $S2$ signal will be activated. Note also that switch 5 has its RR_{nw} bit active, thus avoiding taking the W output port at the current switch, which would lead to an invalid path.

With LBDRe, the SR_h routing algorithm can be applied with no performance degradation. We will demonstrate this affirmation in Section 6.

5 Deadlock-freedom and Connectivity

In this Section we demonstrate that LBDR is deadlock-free and provides connectivity among all the end-nodes. It must be noted that this can also be applied to LBDRe. As it has been shown before, LBDRe embeds LBDR and therefore it inherits all of its properties.

5.1 Deadlock-freedom

LBDR is not restricted to any particular routing algorithm. Instead, it can support any routing algorithm that provides minimal paths for every pair of end-nodes (as we see in Figure 2.e XY is a bad choice as it does not provide connectivity in an irregular topology). However, the applied routing algorithm must ensure deadlock-freedom and LBDR has to maintain such property. LBDR computes the routing bits from the routing restrictions defined by the routing algorithm. The algorithm is deadlock free if no packet crosses a forbidden routing restriction. Therefore, LBDR must ensure that no packet crosses any routing restriction defined by the routing algorithm.

Imagine there is a deadlock in the network induced by a set of packets that are requesting buffers in a cyclic manner. In that situation a packet in a switch sw along the cycle is mapped at a buffer in an input port i and is requesting an output port o for which a routing restriction is defined between i and o. Without lose of generality, consider the input port is S and the output port being requested is W. Hence, a SW routing restriction is defined at switch sw.

As routing restrictions are assigned only to links between switches (links connecting end-nodes are excluded), the given packet has previously been forwarded from a previous switch (sw_p). The output port used to forward the packet at sw_p is N. At this switch the routing bit R_{nw} is set to zero (since there is a SW routing restriction at switch sw). Additionally when routing the packet at switch sw_p the signals N' and W' were active as the packet is now requesting output port W at switch sw. Looking at the LBDR logic for output port N, at switch sw_p the N port can not be selected since none of the outputs of the AND gates will be active (x_1=x_2=x_3=0, see Figure 3). Indeed, the packet is in the NW direction and the Rnw bit is not active. Therefore, this situation can not be induced and thus LBDR is deadlock-free.

Similar conclusions can be obtained when assuming different sets of forbidden routing restrictions (NE, SE and SW).

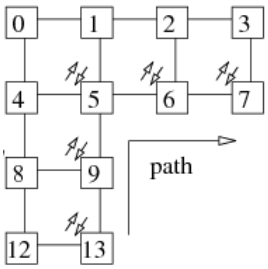


Figure 7. Path taken on boundaries at (p) topology .

5.2 Connectivity

To demonstrate that the mechanism provides connectivity we must first highlight that the routing algorithm implemented by LBDR provides minimal paths and connectivity among all the pairs of end-nodes.

Notice that on each hop a packet performs in the network it gets closer to its destination. From the LBDR logic we can also deduce that non-minimal paths are avoided. Each output port is candidate for being selected only if the destination's distance is reduced along that port. For instance, the N port is eligible only if the packet is in the north direction or in the NW or NE quadrants (the signal N' is active).

Consider the case that a pair of end-nodes can not communicate when using LBDR. In this case, although the routing algorithm provides at least one minimal path to reach the destination end-node the LBDR mechanism fails to provide such path. In that situation, there is a point in the network where either LBDR logic provides a non-minimal path or any of the minimal paths are not eligible.

As an example, consider Figure 7 and the routing unit at switch 8. The packet's destination is switch 3, thus is in the NE quadrant. In that situation the W and S ports are not considered by LBDR since the W' and S' signals are not active. In other words, the packet's destination is neither in the W' nor S' directions. Thus, LBDR avoids non-minimal paths. Therefore, only the N and E directions may be considered. In this case N' and E' signals are activated. In that situation, notice that the N port is eligible only if the R_{ne} bit is active and the E port is eligible only if the R_{en} bit is active. Notice that both bits can not be zero at the same time. In that case there would be no connectivity between switches 8 and 5 and thus, the routing algorithm implemented would not guarantee connectivity. As this situation is not assumed by the routing algorithm, it can not happen, and therefore LBDR guarantees connectivity. Therefore, at least one output port (N or E) will be eligible for routing the packet, getting closer to its destination.

However, a subtle case arises in the boundaries of the topology. Figure 7 shows a p topology and the packet at switch 13 has its destination at the NE quadrant. Switch 13, however, is at the boundaries of the topology. In this case, switch 13 has its R_{en} and R_{ne} bits active. However, its connectivity bit C_e is not active. In this situation the N port is eligible. Notice that the packet will go north until it reaches switch 5 where it will take either N or E.

6 Performance Evaluation

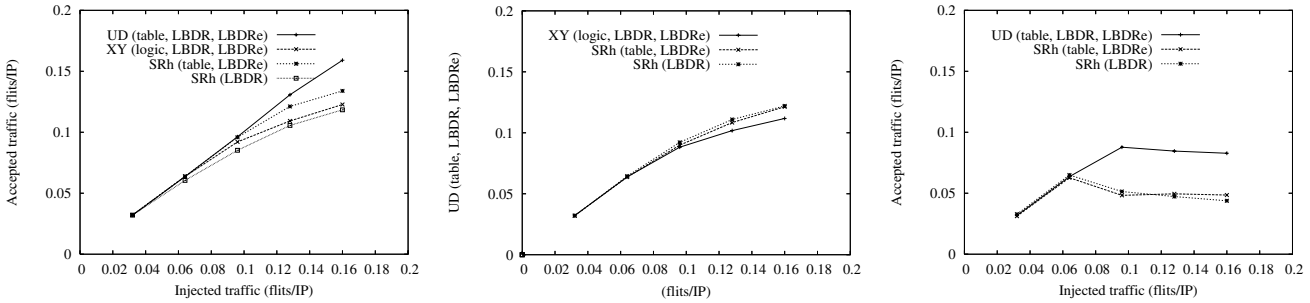
In this Section we evaluate LBDR and LBDRe. Our goal is to evaluate the performance when applied to different topologies/routing algorithms compared with the performance achieved by those routing algorithms when implemented with routing tables. We check if LBDR and LBDRe mimic the performance of routing tables and by how much (and in which circumstances) they lose performance.

We have used noxim [11] to evaluate LBDR and LBDRe. In all simulations wormhole switching is assumed, input port buffers are 4-flit deep, and packets are 32-flit long. Flit size is set to one byte. For the transient state, 40K messages are assumed and results are collected after 40K messages are received. XY, UD, and SR_h routing algorithms have been evaluated in an 8×8 mesh and different irregular topologies: p, q, d, and b topologies (an 8×8 mesh without an 4×4 sub-mesh). Uniform and transpose traffic has been used.

Figure 8 shows the performance (delivered throughput) for some of the analyzed cases. In all the situations the same basic conclusions can be extracted. First, it can be seen that for XY (in 2D mesh) and UD (in 2D mesh and p topology) LBDR mimics the performance achieved with traditional implementation (routing tables). This is achieved because in both cases all the routing restrictions are aligned through the same columns and rows and this permits LBDR to achieve maximum performance.

Second, it can be seen that for SR_h, LBDR achieves different performance numbers. The loss in performance is 15% in transpose traffic and negligible in random traffic.

Finally, Figures 9 and 10 show some interesting performance numbers for SR_h and UD in two different irregular topologies. We can see, that when we apply LBDRe method, as we explained before, for a more complex routing algorithm like SR_h, it performs better than its counterpart on LBDR, even better than UD routing algorithm at low loads, as shown in Figure 9. Both topologies (d and b) have the same shape once they are rotated accordingly, however, we can see that the impact of the routing algorithm can be significant. Indeed, for b topology the achieved performance is higher than in d topology. This yields for interesting future research on finding the best practices in partitioning a NoC.



(a) transpose, 2D mesh (b) random, 2D mesh (c) random, p topology

Figure 8. Performance achieved for different routing algorithms, different topologies, and different traffic patterns.

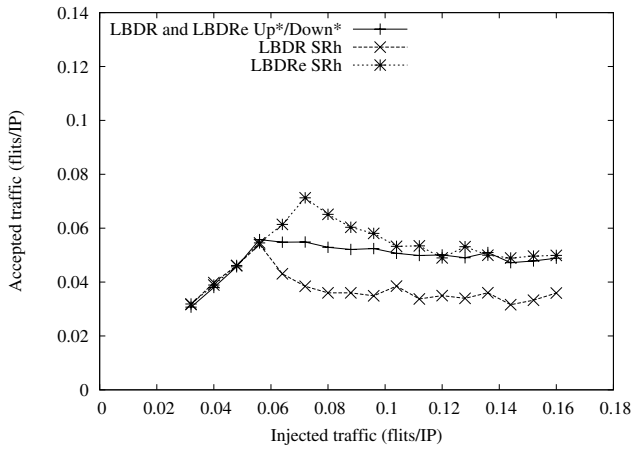


Figure 9. Performance achieved for different routing algorithms and a d topology for both methods. Random traffic.

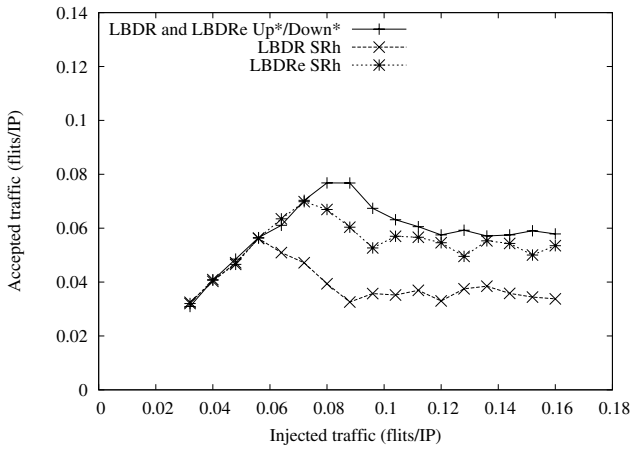


Figure 10. Performance achieved for different routing algorithms and a b topology for both methods. Random traffic.

7 Benefits of LBDR

The benefits of LBDR are important (with no performance degradation at all). With alternative implementations the routing information needs to be stored either at source nodes (with source routing) or at switches (with distributed routing). In both cases routing tables are used. Each table needs as many entries as potential destinations can be addressed from the source or the switch. As the communication pattern in a multi-core is not known in advance the worst case must be assumed. Thus, for a $N \times N$ system up to N^2 entries are needed. Thus, the memory (and area) requirements for alternative implementations grow with system size. With LBDR, however, the requirements are the same regardless of system size. Only three bits for LBDR (7 bits for LBDRe) and a few gates are required per output port. In the same sense power saving can be significant as all the power required by each routing memory is saved.

Moreover, the latency of LBDR could be smaller than the one expected by a table-based implementation. The routing delay of LBDR is given by the time required to compute the relative position of the destination (two comparators in parallel) and the logic associated to each output port (four levels of logic gates). This hardware has an approximate delay of 10 logic gates and very similar to the delay of an XY implementation. As a rough comparison, based on the model proposed in [13, 14] for caches, an access to a routing table should have a delay similar to accessing the data array of a cache. According to [13], for a 0.13 μ m CMOS technology, this value ranges from 0.8 ns for a 4 K cache to more than 2.9 ns for 1 M cache. In the same model a single logic gate has a delay of about 67 ps. Hence, LBDR has a delay of about 0.67 ns. For 90nm technology, gate delays of 9.5ps are available [15], thus the LBDR delay will be 95 ps (not considering wiring delays).

8 Conclusions

In this paper we have presented the LBDR mechanism and its extension, LBDRe. They allow for efficient implementation of most of the existing distributed routing algorithms in suitable topologies for NoCs. For LBDR only two routing bits and one connectivity bit are required along with a small logic per output port. For LBDRe four more bits are required along with the bits existing in LBDR. A bit complex logic than in LBDR method is required but it ensures better performance. LBDR mimics the performance achieved by XY and UD routing algorithms when implemented using routing tables. For more sophisticated routing algorithms, like SR_h, the LBDRe method may be used.

Although not evaluated we have analyzed the extension of LBDR with additional visibility routing bits and obtained no performance gains with any routing algorithm. Therefore, LBDRe, provides enough visibility to extract the full potential of any routing algorithm on an irregular topology. Also, we would like to point that although LBDR loses some performance with some routing algorithms it is much more attractive than LBDRe, due to its simplicity. Also, with a proper routing algorithm (XY and/or UD) the performance penalty is eliminated.

As future work, the applicability of LBDR and LBDRe for chip/system virtualization is meant to be deeply analyzed. Some example of this analysis could be NoC partitioning (we want to assure coherency and isolation for every part, e.g. different running applications) or dealing with failures.

Further studies of area, power and delay reductions (with much more detail) are also on their way. We are currently working on designing and synthetizing the LBDR solution to compare it with a table-based solution.

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", in *IBM Journal of Research and Development*, Volume 49, Number 4/5, 2005.

[3] J. Van Leeuwen and R.B. Tan, "Interval routing", in *The Computer Journal*, 30(4):298-307.

[4] M.E. Gómez et al, "A Memory Effective Routing Strategy for Regular Interconnection Networks", in *International Parallel and Distributed Processing Symposium (IPDPS)*, 2005.

[5] M. Palesi, S. Kumar, R. Holsmark, "A Method for Router Table Compression for Application Specific Routing in Mesh Topology NoC Architecture," in *International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS)*, 2006.

[6] 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.

[7] D. Gelernter, "A DAG-based algorithm for prevention of store-and-forward deadlock in packet networks," in *IEEE Transactions on Computers*, 30(10):709-715, Oct. 1981.

[8] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "Routing Table Minimization for Irregular Mesh NoCs", in *International Conference on Design, Automation and Test in Europe (DATE)*, 2007.

[9] J.C. Sancho, A. Robles, and J. Duato, "A Flexible Routing Schemes for Networks of Workstations", *Third International Symposium on High Performance Computing (ISHPC)*, 2000.

[10] C. Glass and L. Ni, "The Turn Model for Adaptive Routing," in *International Conference on Computer Architecture (ISCA)*, 1992.

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

[12] S. Borkar et. al, "iWarp: An Integrated Solution to High-Speed Parallel Computing," in *Supercomputing Conference*, 1988.

[13] J.E. Wilton, N.P. Jouppi, "An enhanced access and cycle time model for on-chip caches", *Technical Report 93/5*, Western Research Laboratory, 1993.

[14] P. Shivakumar, N. Jouppi, "CACTI 3.0: An integrated cache timing, power and area model", *Technical Report 2001/2*, Western Research Laboratoy, 2001.

[15] Toshiba, product guide 2003, available at `http://www.toshiba.com/taec/components/Generic/cmosasic_tc300prodbroch.pdf`