Logic-Based Distributed Routing for NoCs

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

***Abstract*—The design of scalable and reliable interconnection networks for multicore chips (NoCs) introduces new design constraints like power consumption, area, and ultra low latencies. Although 2D meshes are usually proposed for NoCs, heterogeneous cores, manufacturing defects, hard failures, and chip virtualization may lead to irregular topologies. In this context, efficient routing becomes a challenge. Although switches can be easily configured to support most routing algorithms and topologies by using routing tables, this solution does not scale in terms of latency and area.**

We propose a new circuit that removes the need for using routing tables. The new mechanism, referred to as Logic-Based Distributed Routing (LBDR), enables the implementation in NoCs of many routing algorithms for most of the practical topologies we might find in the near future in a multicore chip. From an initial topology and routing algorithm, a set of three bits per switch output port is computed. By using a small logic block, LBDR mimics (demonstrated by evaluation) the behavior of routing algorithms implemented with routing tables. This result is achieved both in regular and irregular topologies. Therefore, LBDR removes the need for using routing tables for distributed routing, thus enabling flexible, fast and power-efficient routing in NoCs.

I. Introduction

AFTER hitting the power dissipation wall, the computer industry moved to multicore processing chips in order to continue increasing the computing speed while having a bounded power consumption budget. 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 continue for many years, with recent announcements of an 80-core chip [9]. Such a large number of cores requires a high-performance NoC to efficiently interconnect those cores among them and with cache blocks and/or memory controllers.

A 2D mesh topology is usually preferred due to its layout on a 2D surface (Figure 1.a). Also, for routing purposes, logic-based routing (e.g. DOR) is preferred to reduce latency, power, and area requirements for the NoC. However, the high integration scale introduces a number of communication reliability issues. Crosstalk, power supply noise, electromagnetic and intersymbol interference will affect packet transmission. Moreover, manufacturing defects may appear, in the form of defective cores, 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, the presence of such manufacturing defects causes the initial regular topology to become an irregular one. This is the case for topologies shown in Figure 1. In this case, some nodes and links have been disabled. Obviously, other topologies with different shape (p, q, d, b, ...) and number of nodes and links are also possible. Additionaly, in order to exploit the increasing number of cores, and due to the fact that applications are not exposing enough parallelism, virtualization of the chip is becoming a necessity. In a virtualized system, resources are distributted among different guest OS or applications. The network must guarantee traffic isolation between regions, thus leading to irregular subnetworks within the original 2D mesh. Finally, we find irregular topologies in heterogeneous CMPs and MPSoCs where different types of cores exhibit different sizes.

Routing can be implemented as source routing or distributed routing. In source routing, the source node computes the path and stores it in the packet header. Since the header itself must be transmitted through the network, it consumes significant network bandwidth. In distributed routing, however, each switch computes the next link that will be used while the packet travels across the network. The packet header contains only 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 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.

It would be interesting to find an implementation for most practical irregular topologies that allows the use of any distributed routing algorithm without the need for using routing tables. In the present paper we take on this challenge. We pro-

Manuscript submitted: 11-Oct-2007. Manuscript accepted: 19-Nov-2007. Final manuscript received: 23-Nov-2007.
This work was supported by the European Commission in the context of the SARC integrated project #27648 (FP6), by CONSOLIDER-INGENIO 2010 under Grant CSD2006-00046, by CICYT under Grant TIN2006-15516-C04-01, and by Junta de Comunidades de Castilla-La Mancha under Grants PBC-05-005-2 and PCC08-0078.

Authorized licensed use limited to: UNIVERSIDAD POLITECNICA DE VALENCIA. Downloaded on November 4, 2009 at 11:02 from IEEE Xplore. Restrictions apply.

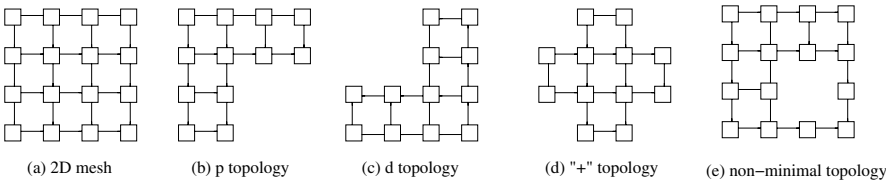


Fig. 1. Examples of topologies (a-d) supported and (e) not supported by LBDR.

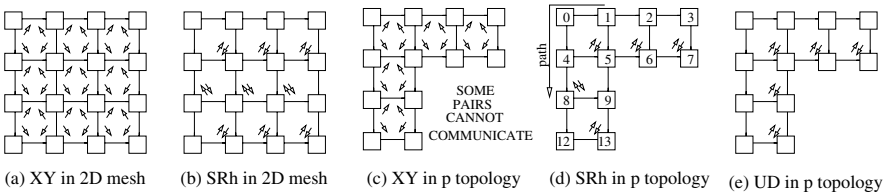


Fig. 2. Examples of routing algorithms (and their routing restrictions) in 2D mesh and *p* topology.

pose a very simple mechanism that removes the routing tables at every switch, thus enabling the implementation of almost 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 block containing several gates.

The rest of the paper is organized as follows. In Section II related work is presented. Then, in Section III the system context for LBDR is described. In Section IV the mechanism is presented. In Section V some evaluation results are presented. The paper is concluded with Section VI.

II. Related Work

Some work on the reduction of memory requirements for routing in NoCs already exists. One solution is Interval Routing [11]. With interval routing, sets of destinations requesting the same output ports are grouped. This method is specifically for regular topologies. The FIR method [6] 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 *street-sign routing* [2]. 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 [8], [3]. In both cases, the destinations are grouped into regions and regions are coded into switches. Although the number of regions grows logarithmically with the number of failures, the number is unbounded and each region implies logic. Another solution for routing table minimization is presented in [1]. In this case logic is used for the general *regular* case, and a deviation routing table 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.

III. System Environment

For the sake of simplicity, we focus on networks with no virtual channel requirements, and we assume wormhole switching (although the proposed method also works for virtual cut-through as well). Messages 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.

All the supported 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 a minimal path defined in the original mesh topology (the topology pictured in Figure 1.a). LBDR can be applied in all the topologies that fulfill this property. As a counter example, Figure 1.e shows a topology with a disabled node and its links. In this case, there are some pairs of end-nodes that cannot communicate through a minimal path defined in the original 2-D mesh case. In this situation, LBDR can not be applied.

Note that topologies with several disabled regions are suitable for LBDR. One example is shown in Figure 1.d. 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 [3], and UD (up*/down*) [4] routing algorithms on a 2-D mesh topology and a *p* topology. Each arrow indicates a routing restriction. Basically, a routing restriction forbids a 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. For instance, at the first (top left-most) switch in Figure 2.a there is a SE restriction [1].

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

Authorized licensed use limited to: UNIVERSIDAD POLITECNICA DE VALENCIA. Downloaded on November 4, 2009 at 11:02 from IEEE Xplore. Restrictions apply.

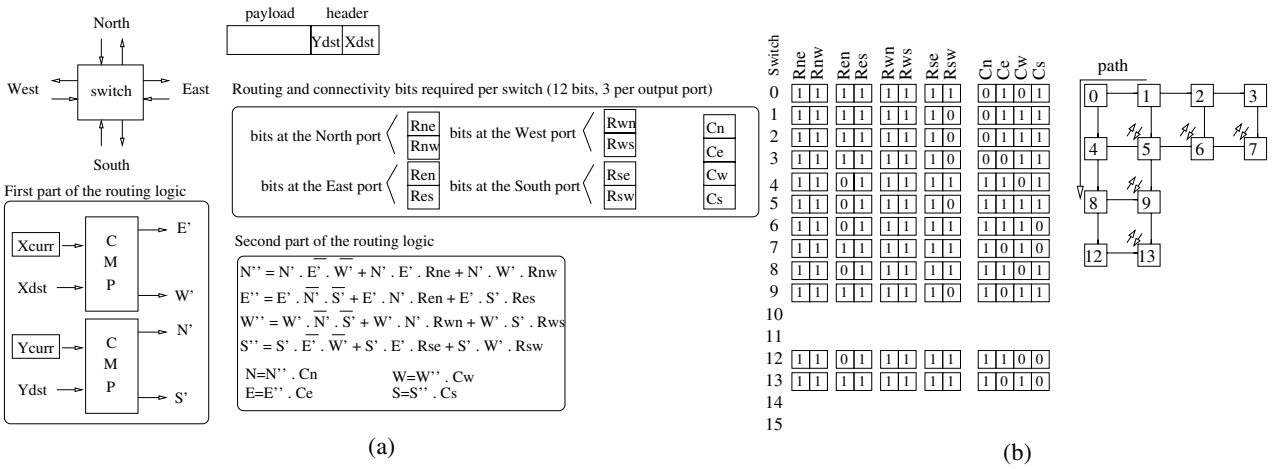


Fig. 3. (a) LBDR method. (b) Example of LBDR for an irregular (*p*) topology. UD routing.

As can be seen in the Figure, the XY routing algorithm is suitable only for the 2D mesh topology. However, topology-agnostic routing algorithms like SR_h, and UD can be applied to any topology.

LBDR is applicable to any routing algorithm that enforces minimal paths for every source-destination pair. This is the case of XY, SR_h, and UD. Other routing algorithms like FX [10] and Turn Model [5] also adhere to these conditions, thus being suitable for LBDR.

IV. LBDR Description

Figure 3.a shows the details of LBDR. The mechanism relies on the use of only three bits per switch output port. Therefore, 12 bits are needed 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.

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 neighbours.

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 later at the next switch take the N port or S port, respectively. In other words, these bits indicate whether packets are allowed to change direction at the next switch. Similarly, 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.

The routing logic is divided in two parts (see Figure 3.a). 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}. From that logic one or two signals may be active (if the packet is in the NW quadrant then N' and W' signals will be 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 is made of four logic units, one for each output port. Each one can be built with only two inverters, four AND gates and one OR gate. As all of them are similar we will describe only one, 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:

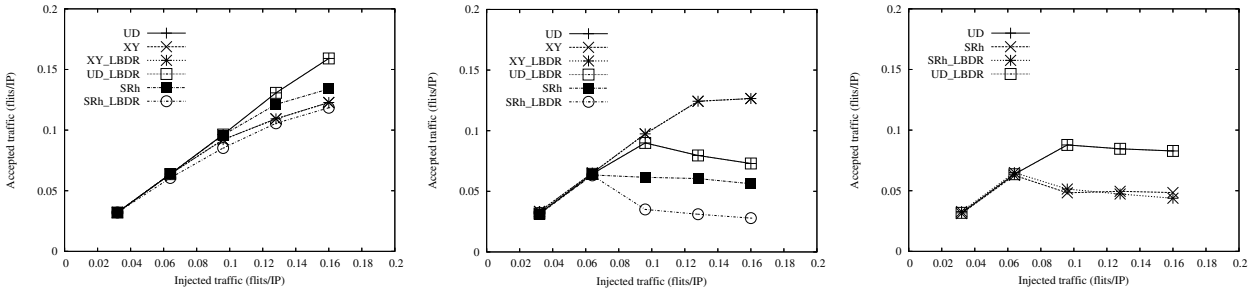
- The packet's destination is on the same column ($N' \times /E' \times /W'$).
- The packet's destination is on the NE quadrant and the packet can take the E port at the next switch through the N port ($N' \times E' \times R_{ne}$).
- The packet's destination is on the NW quadrant and the packet can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

If none of the above conditions is met, then the N port can not be taken for routing the packet. Additionaly, the connectivity bit C_n is inspected in order to filter the N port.

LBDR will mimic performance of 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 3.b shows all the bits for UD in a p topology. Notice that for the path 1-8 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 on 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).

However, there are situations (e.g. more advanced routing algorithms like SR) where LBDR induces some inefficiencies. An example can be seen in Figure 2.d. In this case, like before, switch 1 decides to reject output port S because its R_{sw} bit is reset (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.

Authorized licensed use limited to: UNIVERSIDAD POLITECNICA DE VALENCIA. Downloaded on November 4, 2009 at 11:02 from IEEE Xplore. Restrictions apply.



(a) transpose #1, 2D mesh (b) random #2, 2D mesh (c) random #2, p topology

Fig. 4. Performance achieved for different routing algorithms, different topologies, and different traffic patterns.

Therefore, LBDR reduces adaptiveness. In Section V we will see the impact on performance.

It is important to note that only the routing 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 3.b from switch 13 to switch 7. At switch 13 the signals N' and E' are active. Also, signals N'' and E'' 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 will take east direction. Notice that output port E will never be taken at switches 13 and 9 due to the connectivity bit C_e.

V. Evaluation

In this Section we evaluate LBDR. 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 will check if LBDR mimics the performance of routing tables and by how much (and in which circumstances) it looses performance.

We have used noxim [7] to evaluate LBDR. 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 a 8×8 mesh and a p topology (a 8×8 mesh without the bottom right-most 4×4 submesh). Uniform and bit-reversal traffic has been used.

Figure 4 shows the performance (delivered throughput) for all 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.

VI. Conclusion

In this paper we have presented the LBDR mechanism. It allows for implementing most of the existing distributed routing algorithms in suitable topologies for NoCs. Only two routing bits and one connectivity bit are required along with a small logic block per output port. It mimics the performance achieved by XY and UD routing algorithms when implemented using routing tables.

As future work, the applicability of the LBDR for chip/system virtualization will be deeply analyzed. Also, we have extended the LBDR mechanism to support more complex algorithms, like SR_h, with no performance degradation. Also, demonstrations of deadlock-freedom and connectivity have been obtained (not shown in this paper due to lack of space).

References

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

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

[3] 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.

[4] 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.

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

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

[7] Noxim: NoC simulator, available at http://noxim.sourceforge.net.

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

[9] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, "A 5-GHz Mesh Inte rconnect for a Teraflops Processor", in *IEEE Micro Magazine*, Sept-Oct. 2007, pp. 51-61.

[10] J.C. Sancho, A. Robles, and J. Duato, "A Flexible Routing Scheme for Networks of Workstations," in *High Performance Computing Conference (ISHPC)*, 2000.

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

Authorized licensed use limited to: UNIVERSIDAD POLITECNICA DE VALENCIA. Downloaded on November 4, 2009 at 11:02 from IEEE Xplore. Restrictions apply.