Journal of Systems Architecture 56 (2010) 256–264

Contents lists available at ScienceDirect

Journal of Systems Architecture

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

EDXY – A low cost congestion-aware routing algorithm for network-on-chips

P. Lotfi-Kamran [a,*], A.M. Rahmani [a], M. Daneshtalab [a,b], A. Afzali-Kusha [a], Z. Navabi [a,c]

[a] Nanoelectronics Center of Excellence, School of Electrical and Computer Engineering, University of Tehran, Iran
[b] Computer Systems Laboratory, University of Turku, Turku, Finland
[c] Dept. of Electrical and Computer Engineering, Northeastern University, Boston, USA

ARTICLE INFO

Article history:
Received 27 September 2009
Received in revised form 26 March 2010
Accepted 4 May 2010
Available online 12 May 2010

Keywords:
Network-on-chip
Routing algorithm
Adaptive
Dynamic XY
Bursty traffic
Non uniform traffic
Low latency routing
Link failure tolerant

ABSTRACT

In this paper, an adaptive routing algorithm for two-dimensional mesh network-on-chips (NoCs) is presented. The algorithm, which is based on Dynamic XY (DyXY), is called Enhanced Dynamic XY (EDXY). It is congestion-aware and more link failure tolerant compared to the DyXY algorithm. On contrary to the DyXY algorithm, it can avoid the congestion when routing from the current switch to the destination whose X position (Y position) is exactly one unit apart from the switch X position (Y position). This is achieved by adding two congestion wires (one in each direction) between each two cores which indicate the existence of congestion in a row (column). The same wires may be used to alarm a link failure in a row (column). These signals enable the routing algorithm to avoid these paths when there are other paths between the source and destination pair. To assess the latency of the proposed algorithm, uniform, transpose, hotspot, and realistic traffic profiles for packet injection are used. The simulation results reveal that EDXY can achieve lower latency compared to those of other adaptive routing algorithms across all workloads examined, with a 20% average and 30% maximum latency reduction on SPLASH-2 benchmarks running on a 49-core CMP. The area of the technique is about the same as those of the other routing algorithms.

1. Introduction

Recently on-chip transistor density has increased enabling the integration of dozens of intellectual property cores on a single die to form system-on-chips (SoCs). One byproducts of the greater integration is that for communication in these systems, shared buses should be replaced by interconnection networks. The network-on-chips (NoCs) has been proposed as a new paradigm for realizing complex SoCs [1,2]. NoCs scale better than traditional forms of on-chip interconnections and have better performance and fault tolerant characteristics [2]. Among different possible topologies, the two-dimensional mesh is one of the most common topologies [3,4].

In NoCs, routing algorithms are used to determine the path of a packet from the source to the destination. These algorithms are classified as deterministic and adaptive. The implementations of deterministic routing algorithms are simple but they are not able to balance the load across the links in non-uniform or bursty traffic [5,6]. Adaptive routing algorithms are proposed to address these limitations. By better distributing load across links, adaptive algorithms improve network performance and also provide tolerance if link or router failure occurs. In adaptive routing algorithms, the path of a packet from the source to the destination is determined by the network condition. An adaptive routing algorithm decreases the probability of passing a packet from a congested or mal-function link. Despite its implementation complexity, adaptive routing is attractive for large NoCs especially when these NoCs facing with non-uniform or bursty traffic.

There are a number of routing algorithms which we briefly review those related to the algorithm proposed in this work. In [7], a static routing algorithm for two-dimensional meshes which is called XY is introduced. In this routing algorithm, each packet first travels along the X and then the Y direction to reach the destination. For this method, deadlock never occurs but no adaptivity exists in this algorithm. An adaptive routing algorithm named turn-model is introduced in [8] based on which another adaptive routing algorithm called Odd–Even turn is proposed in [9]. To avoid deadlock, Odd–Even method restricts the position that turns are allowed in the mesh topology. Another algorithm called DyAD is introduced in [10]. This algorithm is a combination of a static routing algorithm called oe-fix, and an adaptive routing algorithm based on the Odd–Even turn algorithm. Depending on the congestion condition of the network, one of the routing algorithms is invoked. Another adaptive routing is hot potato or deflection routing [11,12] which is based on the idea of delivering a packet to an output channel at each cycle. If all the channels belonging to minimal paths are occupied, then the packet is misrouted. When contention

* Corresponding author.
E-mail address: plotfi@computer.org (P. Lotfi-Kamran).

1383-7621/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.sysarc.2010.05.002

occurs and the desired channel is not available, the packet, instead of waiting, will pick any alternative available channels (minimal or non-minimal) to continue moving to the next router; therefore the router does not need buffers. In hot potato routing, if the number of input channels is equal to the number of output channels at every router node, packets can always find an exit channel and they are deadlock free. However, livelock is a potential problem in this routing. Also, hot potato increases message latency even in the absence of congestion and bandwidth consumption. Accordingly, performance of hot potato routing is not as good as other wormhole routing methods [13]. Also, there are adaptive routings for increasing fault tolerance of the on-chip network. Stochastic communication method has been proposed to deal with permanent and transient faults of network links and nodes [14]. This method has the advantage of simplicity and low overhead. The selection of links and of the number of redundant copies to be sent on the links is stochastically done at runtime by the network routers. As a result, the transmission latency is unpredictable and, hence, it cannot be guaranteed. Also, stochastic communication is not efficient in terms of power dissipation and latency.

An adaptive deadlock free routing algorithm called Dynamic XY (DyXY) has been proposed in [15]. In this algorithm, which is based on the static XY algorithm, a packet is sent either to the X or Y direction depending on the congestion condition. It uses local information which is the current queue length of the corresponding input port in the neighboring routers to decide on the next hop. It is assumed that the collection of these local decisions should lead to a near-optimal path from the source to the destination. The main weakness of DyXY is that the use of the local information in making routing decision could forward the packet in a path which has congestion in the routers farther than the current neighbors. This situation could happen when the routing unit is one unit apart from the destination in X or Y dimension. Such non-optimal routing decisions will cause NoC to face with increasing in its network latency. The technique described in [16], may overcome this problem. It uses global information in making a routing decision. The technique requires a mechanism to mix local and global congestion information. This has been obtained at the cost of higher hardware overhead.

In this paper, we propose a technique for solving the problem of the DyXY routing algorithm with little area overhead. In addition, the proposed technique increases tolerance against single link failure compared to the DyXY technique. The rest of this paper is organized as follows: Section 2 describes the basic structure of the XY and DyXY routers. Section 3 describes the proposed routing algorithm and its architecture. Single link failure tolerances of the routing algorithm are compared in Section 4 while experimental results are discussed in Section 5. Finally, the conclusion of the paper is given in Section 6.

2. XY and DyXY routing mechanisms and their limitations

This section describes XY, and DyXY NoC routings and their main limitations.

2.1. XY routing mechanism

Except for the boundary routers, XY routers have five input/output ports (four connected to the neighboring ports and one for the local core). Main architectural elements of an XY router include the input FIFO for each port, route computation unit, VC allocation unit (if any), crossbar control logic, and the crossbar. The packet transmission is performed by dividing it to smaller pieces called flits. A flit enters into the router through one of the ports and is stored in its FIFO. If the flit is a header, indicating the start of a new packet, it proceeds to the routing unit, which determines the output port that the packet should use. The header flit attempts to acquire a channel (maybe virtual) for the next hop. Upon a successful channel allocation, the header flit enters the switch arbitration stage, where it competes for the output port with other flits from the other router input ports. Once the crossbar passage is granted, the flit traverses the switch and enters the channel. Subsequent flits belonging to the same packet can proceed directly to the crossbar and go to the output port. The main architectural element of an XY router is shown in Fig. 1.

2.2. DyXY routing mechanism

The main difference between the XY and the DyXY routers is that, depending on the network condition, the route computation unit may select different paths at different times for the same source and destination pair. For this to happen, a pre-port selection unit is added to the router that based on the current network conditions, selects the best candidate for every adjacent ports (i.e., North vs. East, North vs. West, South vs. East, and South vs. West) and provide route computation unit with this information (Fig. 2). One of the factors for choosing an output port is the number of free buffers at the corresponding input port in the next hop [17]. This technique has been used in the DyXY routing algorithm where the free buffer count at a downstream node is used for congestion estimation. To exchange the count, which can be considered as a stress value, some wires are added between adjacent routers.

Due to the fact that in each node, packets can be routed in both X and Y directions without restriction, this routing algorithm needs a mechanism to guarantee deadlock avoidance. In networks having virtual channels (general case), usually the following method is used to guarantee deadlock avoidance. Virtual channels in Y dimensions are divided into two parts. The network is partitioned into two sub-networks called +X sub-network and −X sub-network each having half of the channels in the Y dimension. If the destination node is to the right of the source, the packet will be routed through the +X sub-network. If the destination node is to the left of the source, the packet will be routed through the −X sub-network. Otherwise that packet can be routed using either sub-network [18]. For networks without virtual channels, other deadlock avoidance strategies should be used (e.g., Odd–Even [19]).

Now, we discuss the weakness of the DyXY algorithm in routing packets from switches whose X position (Y position) is one unit apart from that of the destination. Let us consider the simple

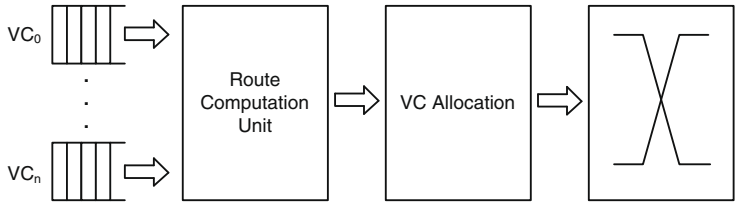


Fig. 1. Structure of an XY router.

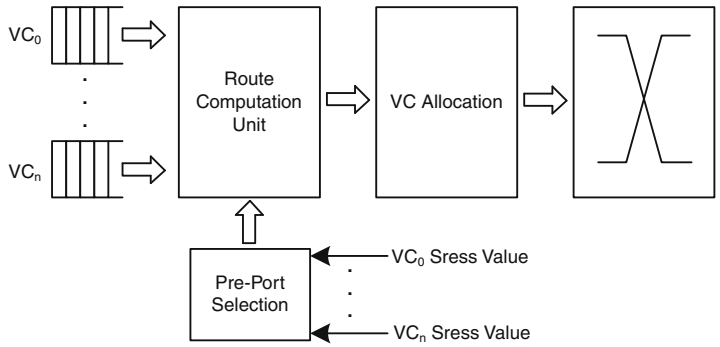


Fig. 2. Structure of a DyXY (adaptive) router.

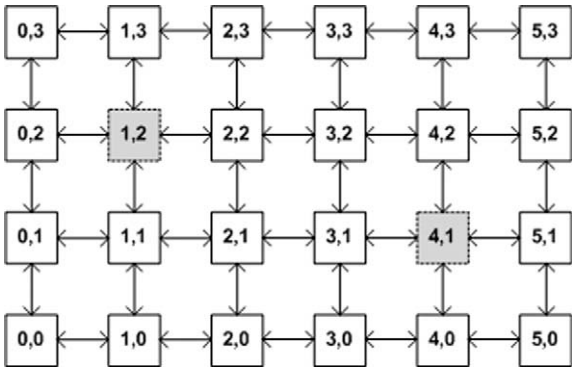


Fig. 3. A simple NoC with mesh structure.

example shown in Fig. 3 where (1, 2) and (4, 1) are the source and destination switches. In the DyXY routing, switch (1, 2) compares the current length of the west queue of switch (2, 2) and the north queue of the switch (1, 1) and the packet is sent to the direction with a less occupied queue. Note that if switch (1, 1) is selected, *the entire path has been determined* and hence switches (2, 1) and (3, 1) are the next hops without any choice. If they are congested, the DyXY algorithm has selected wrong path based on local information. However, if switch (2, 2) is selected, there are other choices for the next hops, and hence, the congestion may be avoided. The wrong path was selected since the decision was based on the local information of switch (1, 1). Note that this undesired effect may occur when routing from a switch whose X (or Y) position are just one unit apart from that of the destination and the congestion is happening further away from the current hop.

3. EDXY routing solution

The objective of the EDXY routing algorithm is to avoid the problem of the DyXY algorithm. This is achieved by using a flag which indicates congestion along the path of a row (or column). This flag propagates in a row (or column) and indicates to the adjacent rows (or columns) that this row (or column) is near saturation and should be avoided. Since congestion flag should propagate along a row (or column), each switch transparently propagates its prior switch congestion flag. Also, each router monitors its input buffer. If the occupied percentage of the buffer is larger than a threshold value, the corresponding switch will activate its congestion flag. The congestion information for the West, East, North, South input port is given to the switches which are to the left, right, up, and down-side of this switch, respectively. To effectively track the congestion condition, two flags are added to each row (or column): one informs left hand side switches of congestion in a right hand side switch and the other one informs right hand side switches of congestion in a left hand side switch in the row. For this purpose, between every two adjacent switches, two congestion wires, one in each direction, are added. The congestion wires are added to the wires used to transmit the free buffer count. The congestion flag transmitted by any switch is obtained by ORing its flag and that of the previous switch (to transparently propagate prior switch congestion flag) in the direction that the flag is transmitted. Therefore, if the flag is active, it shows that either the current or at least one of the previous switches is congested.

In EDXY routing, if a switch is one hop apart from the destination in a row (or column), the congestion flag in the destination column (or row) is used in routing decision. If the congestion flag is one, the algorithm favors the other path to the destination. For the example shown in Fig. 3, the congestion wire in the destination row is passed to switch (1, 2) by switch (1, 1). Therefore, the packet will be routed to switch (2, 2).

In EDXY, every router first looks at the destination address of the received packet. If the router address is not one hop apart from that of the destination in either the X or Y direction, the EDXY algorithm ignores the congestion wire and routes the packet the same way as the DyXY algorithm does. However, if the destination address is just one hop apart from the router in either the X or Y direction, in addition to the current queue length, one of the congestion wires (based on the position of the destination) is also used for routing. We decided to use this congestion value as most significant bit of the stress value. In these cases, we refer to the port which is one hop apart from the destination in either X or Y directions as critical port while the other port is called non-critical port. For the non-critical port, we ignore congestion flag value and a zero is used for the congestion value to favor this port against the other port.

It should be noted that congestion wires may report false positive information because they do not provide information regarding the location of the congestion node in a row (or column). We compared shared congestion wires with the general case in which the position of the congestion node is also propagated in a row or column. The results show only slight differences between these two mechanisms. We proposed shared congestion wires because they are simple and have the desired characteristics for reducing latency.

In cases where the congestion wire should be used in the decision making process, a function of the queue length and the congestion wire value can be used as the stress value. In order to simplify hardware, we used congestion wire as the MSB of the queue length to form the stress value. As we will discuss later, this will enable us improve the single link failure tolerance of the EDXY algorithm.

For the implementation of the EDXY routing algorithm, extra hardware should be added to the DyXY router. This extra hardware

is divided into two parts. An extra unit is needed in each router to drive the congestion wires in four directions. In addition, each router should have extra logic to use the congestion wires in the routing decisions. In each direction, the output congestion wire is set if the input congestion wire due to the congestion in the previous routers in that direction is set or the occupied part of the input buffer for routing in that direction is larger than the threshold value. For implementing this logic, a comparator and an OR gate are used. The comparator compares the queue length with the predefined threshold value. In the case of exceeding the threshold value, the output of the comparator becomes one. The output congestion wire is the OR of the comparator output and the input congestion wire as shown in Fig. 4(a).

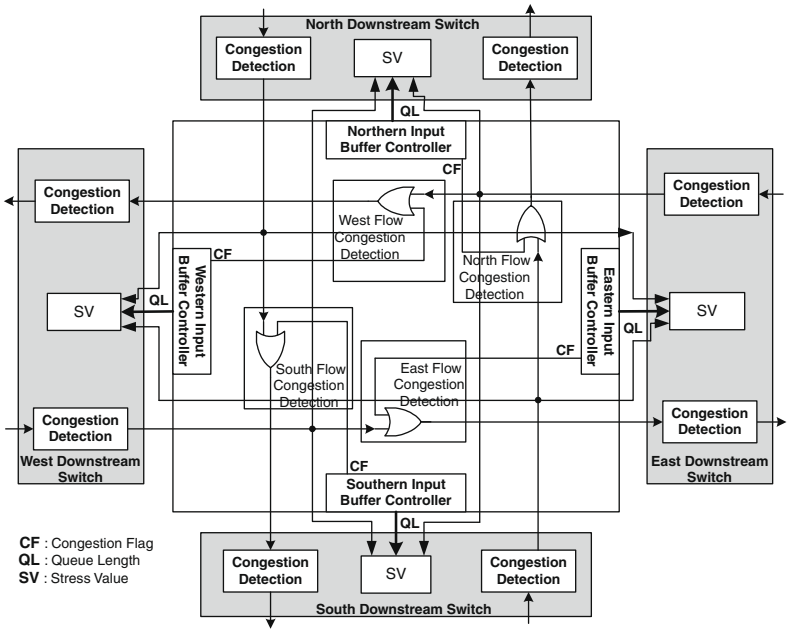
The routing algorithm should also be modified to use the extra information (i.e., congestion wires) in routing decisions. The EDXY routing algorithm is shown in Fig. 4(b). Compared to the DyXY routing architecture, extra hardware (SV) is needed to produce the stress value for two competing ports from their queue length and the values of the congestion wires.

4. Link failure tolerance

The extra wires added to the NoC can be used to empower EDXY to tolerate single link failure. In fact these wires behave as congestion flags in normal conditions and are used to decrease latency of the routing algorithm, while in faulty condition, the role of these wires change and they are used to empower EDXY to route all packets to the destinations. With two modifications to the EDXY routing algorithm, this algorithm can tolerate single link failure (unidirectional and bidirectional) in the network. These modifications do not have any effect on the routing decision in normal conditions.

We assume routers are equipped with test units that signals if links of the router get faulty. When a router detects the failure of one of its links, it will permanently set the output congestion wire corresponding to that link while permanently reset the congestion wire of the links adjacent to the faulty link (Fig. 5(1)). Due to the EDXY routing mechanism, the congestion wire is used as the MSB of the stress value. This will guarantee that a packet will

a



b

$X_{diff} = X_c - X_d$
$Y_{diff} = Y_c - Y_d$
If (one destination port) **Then**
  Use that port to route the packet
Else
  If (ABS(X_{diff}) = 1) **Then**
    If (Y_{diff} > 0) **Then**
      $S\_Value_X = \{I\_queue\_length_X, C\_Wire_{UpToDown}\}$;
    Else
      $S\_Value_X = \{I\_queue\_length_X, C\_Wire_{DowntoUp}\}$;
  Else
    $S\_Value_X = I\_queue\_length_X$;
  End if
  If (ABS(Y_{diff}) = 1) **Then**
    If (X_{diff} > 0) **Then**
      $S\_Value_Y = \{I\_queue\_length_Y, C\_Wire_{RightToLeft}\}$;
    Else
      $S\_Value_Y = \{I\_queue\_length_Y, C\_Wire_{LeftToRight}\}$;
  Else
    $S\_Value_Y = I\_queue\_length_Y$;
  End If
  Use S_Value_X and S_Value_Y to choose the destination port.
End If

Fig. 4. (a) An EDXY switch implementation. (b) EDXY routing algorithm.

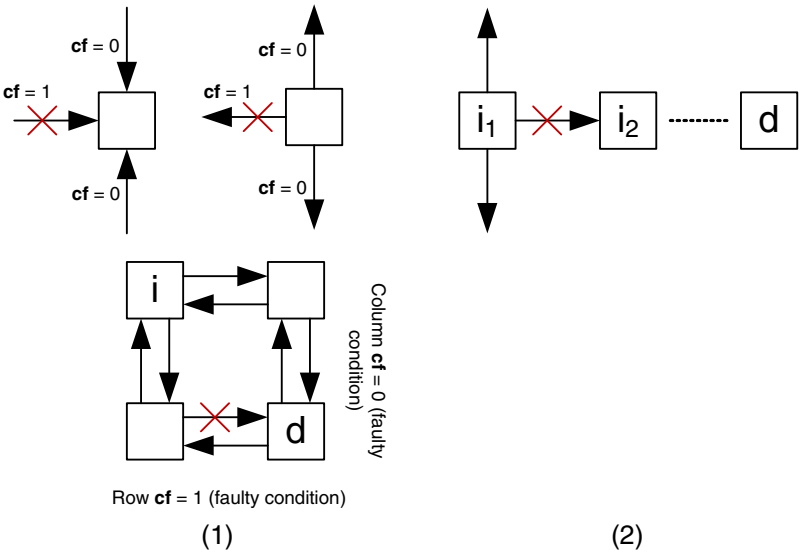


Fig. 5. Two modifications to the EDXY routing algorithm to guarantee single link failure tolerance.

not be passed to the row (or column) when the failed link is before the destination in that row (or column). One corner case that should be considered is the case shown in Fig. 5(1) that the switch has a distance of one to the destination in both the X and Y directions. As shown, for the direction of the failed link, the value of the congestion wire is "1" and for the other direction the value of congestion wire is "0". As such, EDXY guarantees to send the packet through the non-faulty link to the destination.

The other modification to the EDXY routing algorithm is for the source–destination pairs located in a row (or column). In this case, only one path exists between the source and destination and as a result of failed link, this path cannot be used to send the packet to the destination. To overcome this limitation, in a router, if there is just one path to the destination and the link corresponding to the path is failed, the packet will be forwarded to one of the adjacent links as shown in Fig. 5(2). Because we only forward a packet through a non-minimal path in case of a link failure and only for source–destination pairs in a row (or column), the lack of livelock in the routing algorithm is preserved. With these two modification, EDXY routing algorithm can tolerate single link failure in the network. This is especially important in future chip fabrication process in which the failure rate is high. If the NoC chip is fabricated correctly, the added wires are used to decrease network latency, and in the case of a link failure, these wires guarantee the packet transmission in the network.

5. Experimental results

For assessing the efficiency of the proposed routing algorithm, three other routing algorithms were also implemented. These algorithms included the XY, Odd–Even turn-model, and DyXY. A detailed VHDL code for the virtual-channel routers was written and simulations were carried out to determine their latency-throughput characteristics. For all the switches, the data width was set to 32-bits. Each input virtual channel had a buffer (FIFO) with the size of six flits. The congestion threshold value (for EDXY routing) was set to four meaning that the congestion condition was considered when four of six buffer slots were full. In all the simulations, the latency was measured by averaging the latency of the packets when each local core generated 3000 packets. The router used the minimally fully adaptive reserved VC deadlock avoidance technique discussed in [18]. Four synthetic traffic profiles of transpose, uniform random, hotspot 5%, and hotspot 10% and SPLASH-2 benchmark traces were used. Table 1 shows the baseline network configuration, and the variations used in the sensitivity studies.

5.1. First set of experiments

In the first set, 7×7 2D meshes and packets with a length of nine-flits were used. The packet latency for different traffic profiles are shown in Fig. 6.

5.1.1. Transpose traffic profile

In transpose traffic profile, for a $n \times n$ mesh network, a core at position $(i,j)(i,j \in [0,n))$ only sends a data packet to another core at position $(n-1-i,\ n-1-j)$. This traffic pattern is similar to the concept of transposing a matrix [19]. In these simulations, each core generates packets and injects them into the network using the time intervals determined based on the exponential distribution. This traffic profile leads to a non-uniform traffic distribution with heavy traffics for the central nodes of the mesh. Therefore, close to the center of the network hotspots may be created. As the results shown in Fig. 6(a) reveal if the data packet injection rate is very low, hotspots are not created and the routing technique behave similarly. As the injection rate increases and congestion is created in the mesh, the EDXY algorithm leads to smaller average delays.

5.1.2. Uniform random traffic profile

In this traffic profile, each node sends several messages to other nodes in the network where a uniform distribution is used to con-

Table 1
Baseline network configuration and variation.

Characteristics	Baseline	Variations
Topology	7×7 2D Mesh	15×15 2D Mesh
Routing	Minimal, fully-adaptive, reserved VC deadlock avoidance	Odd–Even turn-model
Virtual channels/port	2	0
Flit buffers/VC	6	–
Packet length (flits)	9	15
Traffic workload	Transpose, uniform, hotspot	SPLASH-2 traces
Simulated packets/node	3000	–

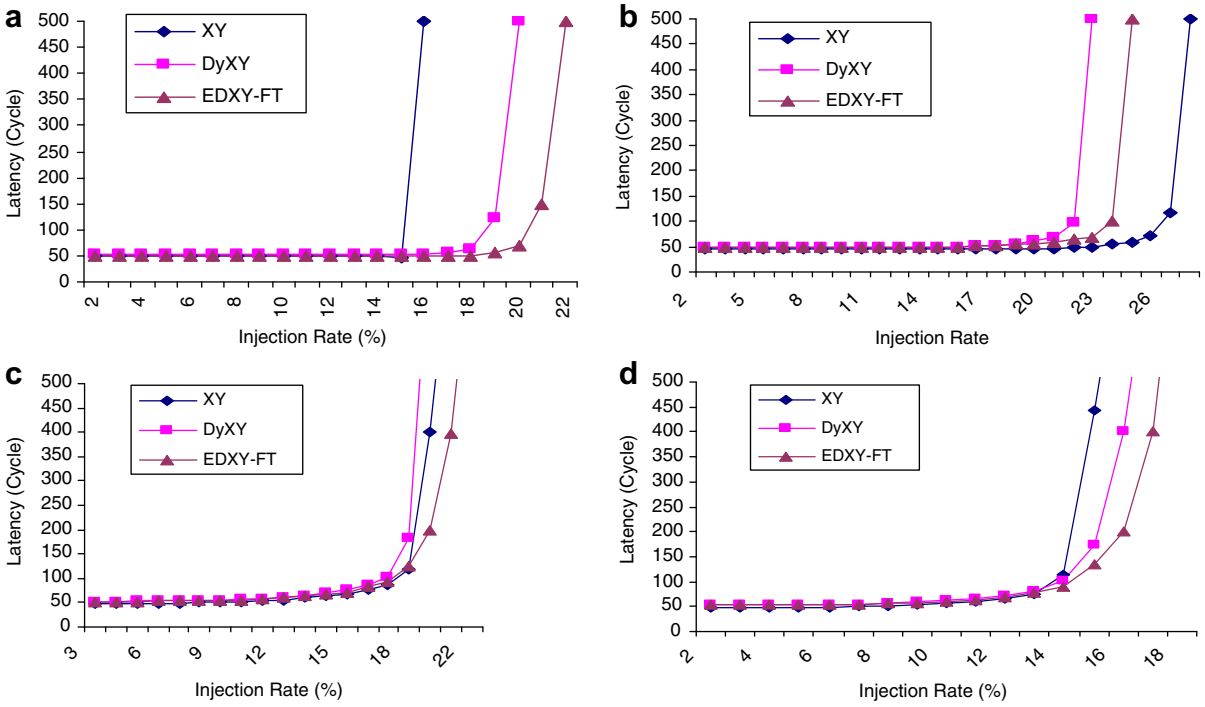


Fig. 6. Latency vs. packet injection rate for EDXY, DyXY, and XY for a 7 × 7 2D mesh for nine-flit packets with virtual channel. (a) Transpose traffic, (b) uniform random traffic, (c) hotspot 5%, and (d) hotspot 10%.

struct the destination set of each message [20]. In Fig. 6(b), the average communication delay as a function of the average message injection rate has been plotted. For this case, the XY technique leads to lower latencies because uniform traffic is balanced under XY routing [16].

5.1.3. Hotspot traffic profiles

The same as uniform random, each node sends several messages to other nodes in the network. But nodes (2, 2), (2, 4), (4, 2), and (4, 4) receives 5% and 10% more packets in hotspot 5% and 10% traffic profiles, respectively. The average communication delays as a function of average packet injection rate for hotspot 5% and 10% traffic profiles have been plotted in Fig. 6(c) and (d), respectively.

5.1.4. SPLASH-2 benchmark traffic

Fig. 7 shows the average packet latency across five SPLASH-2 benchmark traces, normalized to XY. Contention is the cause of significant packet latency in *lu*, *fft*, and *raytrace*; thus adaptive routing has an opportunity to improve performance. Although EDXY provides equal or lower latency than other schemes, EDXY shows the greatest benefit on *raytrace* with 36% reduction in latency. On average EDXY provides a latency reduction of 20% across all benchmarks vs. XY and 12% vs. DyXY.

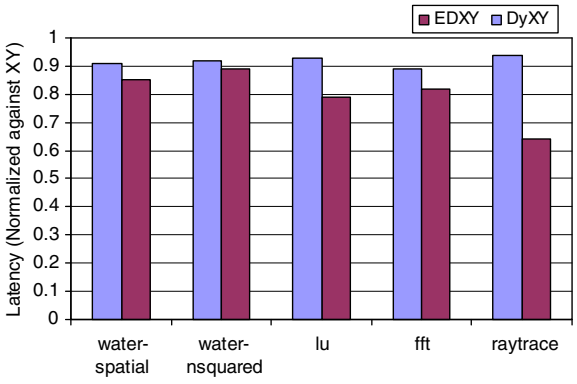


Fig. 7. Average latency across SPLASH-2 benchmarks normalized to latency of XY.

5.2. Second set of experiments

In this section, we change some of the NoC parameters considered in the first set of experiments.

5.2.1. NoC size

Fig. 8 shows the latency vs. the packet injection rate for a 15 × 15 mesh NoC under the transpose traffic profile with nine-flit packets. For this large network, adaptive approaches do not perform as well vs. XY. This is due to the fact that the noise in congestion estimate is increased. But the effect of the noise on DyXY is higher (1%) than EDXY.

5.2.2. Packet length

Fig. 9 shows the latency for long packets (15 flits) in a 7 × 7 2D mesh under different traffic profiles. The average packet latencies

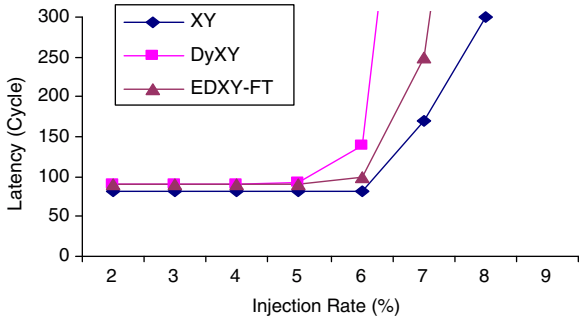


Fig. 8. Latency vs. packet injection rate for 15 × 15 mesh with virtual channel under transpose traffic profile using nine-flit packets.

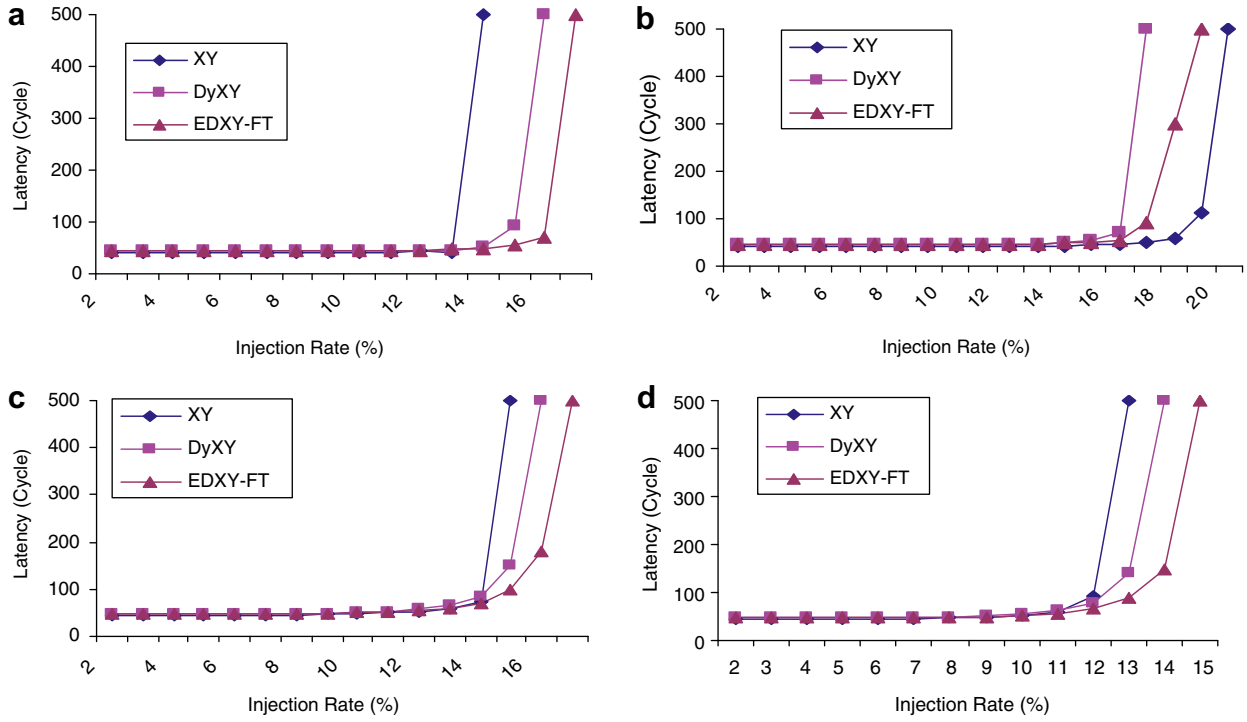


Fig. 9. Latency vs. packet injection rate on a 7 × 7 2D mesh for 15-flit packets with virtual channel. (a) Transpose traffic, (b) uniform random traffic, (c) hotspot 5%, and (d) hotspot 10%.

for all routings are higher than those of short packets. The increased latency is a known characteristic of wormhole routing with long packets. The reason is that for long packets, an imbalance in the resource utilization occurs because packets hold resources over multiple routers. Similar to the case of nine-flit packets, the EDXY technique performs better than DyXY and XY over all traffic patterns except for the uniform traffic profile.

5.2.3. Network without virtual channel

Fig. 10 shows latency vs. packet injection rate for Odd–Even, DyXY, and EDXY routers without virtual channel. In this case, we consider a 5 × 5 2D mesh with five flit packets. The buffer size of each channel is six flits. Odd–Even turn-model [8] is a technique for avoiding deadlock in NoCs without virtual channel. In this section, for the deadlock avoidance in DyXY and EDXY, Odd–Even

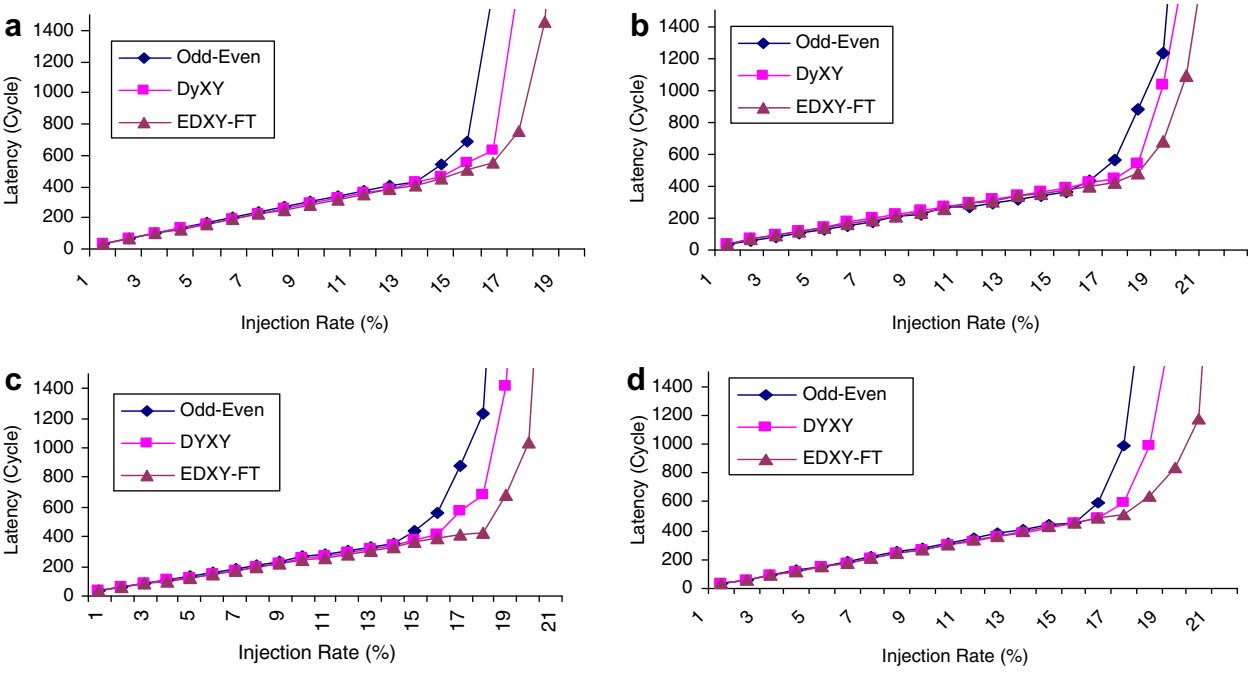


Fig. 10. Latency vs. packet injection rate on a for a 5 × 5 2D mesh without virtual channel. (a) Transpose traffic, (b) uniform random traffic, (c) hotspot 5%, and (d) hotspot 10%.

Table 2
Area comparison of XY, DyXY, and EDXY.

	XY	DyXY	EDXY
Area (μm^2)	86,107	87,881	89,281

Table 3
Power consumption of DyXY, and EDXY routing under GSM voice CODEC (mW).

	DyXY	EDXY
Power consumption (mW)	26.1268	26.1197

turn-model is used. Actually, when the Odd–Even produces more than one output ports, in the Odd–Even router, the port in the Y direction is chosen while in the DyXY and EDXY routers, the stress value is examined to choose one of the ports.

For these simulations, core $(2, 2)$ receives 5% and 10% more packets in hotspot 5% and 10% traffic profiles, respectively. As the results show, EDXY continues to perform better than other routing algorithms in a network without virtual channel with transpose, uniform, hotspot 5%, and hotspot 10% traffic profiles.

5.3. Hardware overhead

To evaluate the area overhead of the proposed algorithm, we synthesized the VHDL reference model with Synopsys Design Compiler using a standard cell CMOS library. For all switches, the data width was set to 32 bits (flit size), and each channel had two virtual channels with a buffer size of six flits. In order to achieve better performance/power characteristics, the FIFOs were implemented using registers. Table 2 compares the areas of the switches. As the figures reveal, the area overhead of the EDXY compared to that of the XY (DyXY) is 3.6% (1.5%).

5.4. Power dissipation

The power dissipation of EDXY and DyXY routing algorithms were calculated and compared under a GSM voice CODEC [21] using Synopsys PrimePower. The GSM voice CODEC was partitioned into nine cores. The communication traces between the cores were recorded for an input voice stream of 500 frames (10 s of voice). The cores were mapped manually onto a 7×7 mesh NoC in the same way as that of Ref. [22]. The CODEC cores generated packets based on the recorded communication traces. The other cores in the NoC generated packets based on the uniform random traffic profile with the same average flit injection rate as the CODEC cores. The typical clock of 1 GHz is applied to the system. Since the post synthesis simulation is very slow, switch $(3, 3)$ which is close to the center of a 7×7 2D mesh was synthesized while for the other switches, the RTL models were used. The results for the power consumption are given in Table 3. As observed from this table, the power consumptions of both techniques are about the same.

6. Conclusions

In this paper, an enhanced dynamic routing algorithm, called EDXY, was proposed. It is congestion-aware and more link failure tolerant compared to the DyXY routing technique. The algorithm improved the DyXY routing algorithm. In this technique, two congestion wires were added to the router architecture to flag the row or column congestion further away from the current switch. This enabled avoiding the congested path, and thus decreasing the latency of the algorithm. The same wires were used to indicate a faulty link in a possible path for a number of source–destination pairs during a link failure. Simulations were performed for EDXY, XY, DyXY, and Odd–Even routing algorithms under several traffic profiles. The results showed lower latencies for the proposed algorithm for high packet injection rates where the congestion in the NoC occurs. The area increase of the proposed switch compared to those of the XY and DyXY was negligible.

References

[1] L. Benini, G.D. Micheli, Networks on chips: a new SoC paradigm, IEEE Computer 35 (January) (2002) 70–78.
[2] W.J. Dally, B. Towles, Route packets, not wires: on-chip interconnection networks, in: Proceedings of the Design Automation Conference, 2001, pp. 684–689.
[3] K. Sankaralingam, R. Nagarajan, P. Gratz, R. Desikan, D. Gulati, H. Hanson, C. Kim, H. Liu, N. Ranganathan, S. Sethumadhavan, S. Sharif, P. Shivakumar, W. Yoder, R. McDonald, S. Keckler, D. Burger, The distributed microarchitecture of the TRIPS prototype processor, in: International Symposium on Microarchitectures, 2006, pp. 480–491.
[4] S. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, P. Iyer, A. Singh, T. Jacob, S. Jain, S. Venkataraman, Y. Hoskote, N. Borkar, An 80-tile 1.28 TFLOPS network-on-chip in 65 nm CMOS, in: IEEE International Solid State Circuits Conference, February 2007, pp. 98–99.
[5] D. Bertsekas, R. Gallager, Data Networks, Prentice Hall, 1992.
[6] W.J. Dally, B. Towles, Principles and Practices of Interconnection Networks, Morgan Kaufmann, 2004.
[7] Intel Corporation, A touchstone delta system description, in: Intel Advanced Information, 1991.
[8] C.J. Glass, L.M. Ni, The turn model for adaptive routing, Journal of the ACM 41 (September) (1994) 874–902.
[9] G.M. Chiu, The odd–even turn model for adaptive routing, IEEE Transactions on Parallel and Distributed Systems 11 (July) (2000) 729–738.
[10] J.C. Hu, R. Marculescu, DyAD – smart routing for networks-on-chip, in: Proceedings of the Design Automation Conference, 2004, pp. 260–263.
[11] E. Nilsson, M. Millberg, J. Öberg, A. Jantsch, Load distribution with the proximity congestion awareness in a network on chip, in: Proceedings of the Design Automation and Test in Europe, 2003, p. 11126.
[12] U. Feige, P. Raghavan, Exact analysis of hot-potato routing, in: Proceedings of the 33rd Annual Symposium on Foundations of Computer Science, 1992, pp. 553–562.
[13] J. Duato, S. Yalamanchili, L. Ni, Interconnection Networks an Engineering Approach, IEEE Computer Society Press, 1997.
[14] T. Dumitras, R. Marculescu, On-chip stochastic communication, in: Proceedings of the Design Automation and Test in Europe, 2003, pp. 10790.
[15] M. Li, Q.-A. Zeng, W.-B. Jone, DyXY – a proximity congestion-aware deadlock-free dynamic routing method for network on chip, in: Proceedings of the Design Automation Conference, 2006, pp. 849–852.
[16] P. Gratz, B. Grot, S.W. Keckler, Regional congestion awareness of load balance in network-on-chips, in: International Symposium on High Performance Computer Architecture, February 2008, pp. 203–214.
[17] J. Kim, D. Park, T. Theocharides, N. Vijaykkrishnan, C.R. Das, A low latency router supporting adaptivity for on-chip interconnects, in: International Symposium on Computer Architecture, 2005, pp. 150–161.
[18] L.M. Ni, P.K. McKinley, A survey of wormhole routing techniques in direct networks, IEEE Computer (1993) 62–76.
[19] C.J. Glass, L.M. Ni, The turn model for adaptive routing, in: Proceedings of the Symposium on Computer Architecture, May 1992, pp. 278–287.
[20] X. Lin, L.M. Ni, Multicast communication in multi-computer networks, in: IEEE Transactions on Parallel and Distributed Systems, 1993, pp. 1105–1117.
[21] M.T. Schmitz, B.M. Al Hashimi, P. Eles, System Level Design Techniques for Energy Efficient Embedded Systems, Kluwer Academic Publishers, 2004.
[22] D. Wu, B.M. Al Hashimi, M.T. Schmitz, Improving routing efficiency for network-on-chip through contention-aware input selection, in: Proceedings of the 2006 Conference on Asia South Pacific Design Automation, 2006, pp. 36–41.

Pejman Lotfi-Kamran received his B.Sc. and M.Sc. degrees in computer engineering from University of Tehran in 2002 and 2005, respectively. His research interest includes various aspects of computer architecture including multi-core architectures, power efficient architectures, service-oriented architectures, and interconnection network. He published dozens of papers in prestigious journals and conferences. Pejman is a student member of IEEE and ACM.

Amir-Mohammad Rahmani received B.S. degree from Mashhad Branch, Azad University, Iran, in 2006, and M.S. degree from the University of Tehran, Tehran, Iran, in 2009, both in Computer Engineering. He is currently pursuing his Ph.D. in Computer Systems Laboratory, University of Turku, Finland. His research interests include Low-Power Design, Network-on-chips, Multi-Processor System-on-chip, and 3D ICs.

Masoud Daneshtalab received his Master's degree in computer architecture from School of Electrical and Computer Engineering, University of Tehran in 2006. Since autumn 2008 he has been working in the Computer Systems laboratory, University of Turku and from May 2009 he is a doctoral candidate of Graduate School in Electronics, Telecommunications and Automation (GETA). He is expected to get his PhD degree in Jan 2011. He has expertise in on/off-chip interconnection networks, multiprocessor architectures, network-on-chips (NoC), and low-power digital design. His PhD thesis is focused on topology formation and routing protocol in 2-D and 3-D On-chip Networks. Masoud is a member of IEEE and has published more than 40 refereed international journals and conference papers.

Ali Afzali-Kusha (SM' 06) received his B.Sc., M.Sc., and Ph.D. degrees in Electrical Engineering from Sharif University of Technology, the University of Pittsburgh, and the University of Michigan in 1988, 1991, and 1994, respectively. From 1994 to 1995, he was a Post-Doctoral Fellow at the University of Michigan. Since 1995, he has been with the University of Tehran, where he is currently a Professor in the School of Electrical and Computer Engineering and the Director of the Low-Power High-Performance Nanosystems Laboratory. Also, while on a research leave from the University of Tehran, he was a Research Fellow at the University of Toronto and the University of Waterloo in 1998 and 1999, respectively. His current research interests include low-power high-performance design methodologies from the physical design level to the system level for the nanoelectronics era. Dr. Afzali-Kusha is a senior member of IEEE.

Zainalabedin Navabi, Ph.D., is professor of electrical and computer engineering at University of Tehran. Dr. Navabi has worked in the design, definition, and implementation of hardware description languages and the synthesis and testing of digital systems. He has developed and supervised the development of many HDL-related software packages and tools, and has directed projects in VLSI design, test synthesis, simulation, synthesis, and other aspects of digital system design automation. Dr. Navabi is a member of ACM, IEEE, and IEEE Computer Society, and is an active participant in the IEEE DASC committee that sets standards related to hardware description languages.