A Methodology for Design of Application Specific Deadlock-free Routing Algorithms for NoC Systems

Maurizio Palesi
DIIT, Univ. of Catania, Italy
mpalesi@diit.unict.it

Rickard Holsmark
Jönköping University, Sweden
hori@ing.hj.se

Shashi Kumar
Jönköping University, Sweden
kush@ing.hj.se

ABSTRACT

In this paper, we present a methodology to specialize the routing algorithm in routing table based NoC routers. It tries to maximize the communication performance while ensuring deadlock free routing for an application. We demonstrate through analysis that routing algorithms generated by our methodology have higher adaptiveness as compared to turn-model based deadlock free routing algorithms for a mesh topology NoC architecture. Performance evaluation is carried out by using a flit-accurate simulator on traffic scenarios generated by both synthetic and real applications. The routing algorithms generated by the proposed methodology achieve an improvement in delay close to 50% and 30% over deterministic XY routing algorithm and adaptive Odd-Even routing algorithm respectively.

Categories and Subject Descriptors

B.4.3 [**Input/Output and Data Communications**]: Interconnections—*Topology*; J.6 [**Computer-Aided Engineering**]: Computer-aided design

General Terms

Networks, Algorithms, Performance

Keywords

Networks on Chip, adaptive routing, deadlock-free routing, application specific

1. INTRODUCTION

In just the last few years Network on Chip (NoC) has emerged as a dominant paradigm for synthesis of multi-core SoCs. A large number of different NoC architectures have been proposed by different research groups [1, 2, 3, 5, 9, 14] based on this paradigm. The proposed architectures differ in many aspects like topology and routing algorithms used in the underlying on-chip communication network. Fixed tile size based mesh topology is favored by many research groups because of its layout efficiency and the resulting electrical properties of the signals.

CODES+ISSS'06, October 22–25, 2006, Seoul, Korea.

It is now possible to envision a scenario in which a mesh topology NoC chip, populated with an application area specific set of cores, will be available as off the shelf standard product. Such a chip will have the potential of high volume of production to justify its large non-recurring expenses. One can easily imagine such a chip for multi-media processing area. Such a chip should implement an adaptive routing algorithm for on-chip communication in order to provide good performance in the presence of traffic variations within an application and among applications in a specific area. However, adaptive routing algorithms, if not designed carefully, have a danger of causing traffic deadlocks. Wormhole switching technique used in communication networks is more efficient than store and forward technique and is therefore proposed by several researchers as the most suitable for on-chip communication [3, 9, 10, 14]. However, this technique is more prone to deadlocks than other switching techniques. Many deadlock free routing algorithms have been proposed for mesh topology networks in literature [4, 6, 7, 8, 9]. In most of these algorithms freedom from deadlocks is achieved at a high loss of adaptivity. Odd-Even routing algorithm [4] provides deadlock free routing in a homogeneous mesh topology NoC architecture. A limitation of the Odd-Even routing algorithm is that it can not ensure deadlock freedom for a irregular mesh topology in which cores could occupy more than one tile. Bolotin *et al.* [3] propose XY routing extended with hard coded paths for deadlock free communication in an irregular mesh topology. A non-minimal deadlock free routing algorithm is described for a irregular mesh topology NoC with regions in [8]. This algorithm is biased in favor of some area of the network as compared to some other areas.

Duato has proposed a general theory to develop adaptive deadlock free routing algorithms for communication networks which use wormhole switching technique [6]. Duato's method is based on generating a *Channel Dependency Graph* (*CDG*), in which every channel is a node and there is a directed edge from a node i to j if channel j can be used after channel i for some communication among resources in the network. A cycle in the *CDG* indicates a possibility of a deadlock. Duato's method restricts some combinations of channels so that cycles could be avoided in *CDG*. Such a routing algorithm can be implemented using routing tables inside routers in the network [2].

Duato's method takes only the network topology as input and generates many routing algorithms which will work for all possible communication traffic situations in the network. This method can be used for generating deadlock free routing algorithms for both regular and irregular networks. One can view all minimal adaptive deadlock free routing algorithms for mesh topology NoC, like Odd-Even routing algorithm, as specific instances of routing algorithms generated by Duato's method.

A NoC system, which is specialized for a specific application or for a set of concurrent applications, can be considered as a semi-static system. We can have the information about the set of pairs of cores which communicate and other pairs which never communicate. But it may not be possible to know the dynamic variations in the communication traffic among the pairs. This information about communication pairs can be useful for generating deadlock free algorithms which are more adaptive than algorithms where this information was not available or used. We call algorithms using this information as *Application Specific Routing Algorithms* (APSRAs).

In this paper, we extend Duato's theory and present a method to generate routing algorithms for communication networks when the communication graph of the application is known. We apply the extended method to generate a routing algorithm for a mesh topology network. We have evaluated and compared the performance of APSRAs with XY and Odd-Even routing algorithms through modeling and simulation. These results demonstrating advantages of APSRAs are presented in the paper.

2. TERMINOLOGY AND DEFINITIONS

In this section we define the concept of *Application Specific Routing*. In an embedded system scenario the communication traffic between different cores of a system-on-a-chip is usually well characterized. In particular, after the task mapping phase of the NoC design flow, we have a complete knowledge about the pairs of cores which communicate and other pairs which never communicate. This additional information can be exploited to design an application specific routing algorithm which is highly adaptive and is also deadlock free.

Given a directed graph $G(V,E)$ where V is the set of vertices and E is the set of edges, we indicate with $e_{ij} = (v_i, v_j)$ the directed arc from vertex v_i to vertex v_j. Given an edge $e \in E$ we indicate with $src(e)$ and $dst(e)$ respectively the source and the destination vertex of the edge (e.g., $src(e_{ij}) = v_i$ and $dst(e_{ij}) = v_j$).

Definition 1. A *Communication Graph* $CG = G(T,C)$ is a directed graph where each vertex t_i represents a task, and each directed arc $c_{ij} = (t_i, t_j)$ represents the communication from t_i to t_j.

Definition 2. A *Topology Graph* $TG = G(P,L)$ is a directed graph where each vertex p_i represents a node of the network, and each directed arc $l_{ij} = (p_i, p_j)$ represents a physical unidirectional channel (link) connecting node p_i to node p_j.

Definition 3. A *Mapping Function* $M : T \rightarrow P$ maps a task $t \in T$ on a node $p \in P$.

Let $L_{in}(p)$ and $L_{out}(p)$ respectively be the set of input channels and output channels for node p. Mathematically:

$$L_{in}(p) = \{l |\ l \in L \wedge dst(l) = p\}$$
$$L_{out}(p) = \{l |\ l \in L \wedge src(l) = p\}.$$

Definition 4. A *Routing Function* for a node $p \in P$, is a function $R(p) : L_{in}(p) \times P \rightarrow \wp(L_{out}(p))$. $R(p)(l,q)$ gives the set of output channels of node p that can be used to send a message received from the input channel l and whose destination is $q \in P$.

The $\wp$ indicates a power set. We indicate with R the set of all routing functions: $R = \{R(p) :\ p \in P\}$.

Definition 5. Given a communication graph $CG(T,C)$, a topology graph $TG(P,L)$, and a routing function R, there is an *application specific direct dependency* from $l_i \in L$ to $l_j \in L$ iff

$$dst(l_i) = src(l_j) \quad (1)$$
$$\exists\, c \in C :\ l_j \in R(dst(l_i))(l_i, M(dst(c))) \quad (2)$$

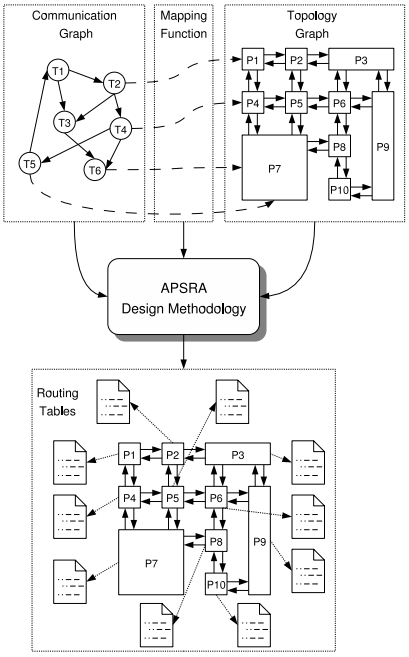


Figure 1: Overview of the APSRA design methodology.

Condition (1) states that there exists a possibility for a message to use l_j immediately after l_i. Condition (2) states that there exists a communication that will actually use l_j immediately after l_i.

Definition 6. An *Application Specific Channel Dependency Graph* $ASCDG(L,D)$ for a given CG, a topology graph TG, and a routing function R, is a directed graph. The vertices of *ASCDG* are the channels of TG. The arcs of *ASCDG* are the pair of channels (l_i, l_j) such that there is an application specific direct dependency from l_i to l_j.

Note that there will be no cycles of length one in *ASCDG* if we assume unidirectional channels.

THEOREM 1. *A routing function R for a topology graph TG and for a communication graph CG is deadlock-free if there are no cycles in its application specific channel dependency graph ASCDG.*

PROOF. The *ASCDG* is a sub-graph of Duato's channel dependency graph (*CDG*) [6]. Two cases need to be considered.

Case 1: Both *ASCDG* and corresponding *CDG* are acyclic. In this case the proof follows the proof of Duato's theorem [6].

Case 2: *ASCDG* is acyclic but corresponding *CDG* has cycles. In each of these cycles in *CDG* there will exist an arc linking two channels l_i and l_j such that there exists no communication pair which can use l_i followed by l_j. We call such cycles as *false cycles* which can be ignored for analysis for deadlock freedom. The resulting *CDG* will then be acyclic. □

3. APSRA DESIGN METHODOLOGY

An overview of the APSRA design methodology is depicted in Figure 1. The inputs are the communication graph CG, the topology graph TG and a mapping function M. The outputs are the routing tables for each node of TG.

Theorem 1 gives a sufficient but not necessary condition for an adaptive routing function R to be deadlock-free. If the application specific channel dependency graph *ASCDG* is acyclic then R

is deadlock-free, otherwise we cannot say anything about the deadlock freeness of R.

The basic idea of APSRA is that a cycle in *ASCDG* can be broken by restricting the routing function of some node while ensuring destination reachability of each communication pair. In this section we present a heuristic to break cycles in the *ASCDG* in such a way as to minimize its impact on the avearge degree of adaptiveness of R. Before we start, some definitions are needed.

Definition 7. A *Path* from a node p_s to a node p_d is a succession of channels $\{l_1, l_2, \ldots, l_n\},\ l_i \in L$ such that:

$$dst(l_i) = src(l_i + 1),\ i = 1, 2, \ldots, n-1,$$
$$p_s = src(l_1),\ p_d = dst(l_n).$$

Given a communication $c \in C$ we denote with $\Phi(c)$ the set of all permissible minimal paths from node $M(src(c))$ to node $M(dst(c))$. We indicate with $\phi_i(c)$ the i-th path of $\Phi(c)$. For each edge d of the *ASCDG* let $A(d)$ be the set of pairs (c, j) where c is a communication whose j-th path contains both channels associated to d. More precisely

$$A(d) = \{(c, j) |\ c \in C,\ j\ \in \mathbb{N}\ s.t.\ src(d) \in \phi_j(c) \wedge dst(d) \in \phi_j(c)\}.$$

Definition 8. Given a routing function R and a communication $c \in C$ the degree of adaptiveness for c is:

$$\alpha(c) = \frac{|\Phi(c)|}{TMP(c)},$$

where $TMP(c)$ represents the total number of minimal paths from node $M(src(c))$ to node $M(dst(c))$.

For mesh based topologies this number is $(dx+dy)!/dx!dy!$ where dx and dy represent the distance in x direction and y direction between the source node and the destination node resepctively.

Definition 9. The average degree of adaptiveness α is the average of the degree of adaptiveness for all the communications.

$$\alpha = \frac{1}{|C|} \sum_{c \in C} \alpha(c).$$

3.1 Main Algorithm

Given a communication graph CG, a topology graph TG and a mapping function M the APSRA methodology can be summarised as follows:

1. Let R be a minimal fully adaptive routing function.
2. Build the *ASCDG* relative to R, CG, TG and M.
3. If *ASCDG* is acyclic then extract routing tables (cf. Section 3.4) and stop.
4. Extract a cycle from *ASCDG*.
5. Use an heuristic to cut an edge (i.e., remove a dependency) of the cycle and update R (cf. Section 3.2).
6. Goto 2.

3.2 Cutting Edge with Minimum Loss

Let $D_c = \{d_1, d_2, \ldots, d_n\} \subseteq D$ be a cycle in the $ASCDG(L, D)$. To break the cycle we have to remove a dependency d_i that means make $A(d_i) = \emptyset$.

To make $A(d_i) = \emptyset$ we have to restrict the number of admissible paths for some communications. This however has an impact on the degree of adaptiveness of the routing function. The heuristic has to select the dependency d_i to be removed in such a way to minimise the impact on the degree of adaptiveness.

Let α be the current degree of adaptiveness and α_d the degree of adaptiveness when we remove a dependency $d \in D_c$. The objective is to minimise the difference $\alpha - \alpha_d$, or equivalently maximise α_d.

$$\begin{aligned}
\max_{d \in D_c} \alpha_d &= \max_{d \in D_c} \frac{1}{|C|} \sum_{c \in C} \frac{|\Phi_d(c)|}{TMP(c)} = \\
&= \max_{d \in D_c} \frac{1}{|C|} \sum_{c \in C} \frac{|\Phi(c) \setminus \{a \in A(d) : a.c = c\}|}{TMP(c)} \\
&= \max_{d \in D_c} \frac{1}{|C|} \sum_{c \in C} \frac{|\Phi(c)| - |\{a \in A(d) : a.c = c\}|}{TMP(c)} \\
&= \max_{d \in D_c} \frac{1}{|C|} \left(\sum_{c \in C} \frac{|\Phi(c)|}{TMP(c)} - \sum_{c \in C} \frac{|\{a \in A(d) : a.c = c\}|}{TMP(c)} \right)
\end{aligned}$$

That is equivalent to:

$$\min_{d \in D_c} \sum_{c \in C} \frac{|\{a \in A(d) : a.c = c\}|}{TMP(c)} = \min_{d \in D_c} \sum_{a \in A(d)} \frac{1}{TMP(a.c)}.$$

In short, the heuristic states that to minimise the impact on adaptiveness we have to select a dependency $d \in D_c$ to be removed which satisfy the following reachability constraint:

$$\bigwedge_{(c,j) \in A(d)} |\Phi(c)| > 1, \tag{3}$$

and minimise the quantity:

$$\sum_{(c,j) \in A(d)} \frac{1}{TMP(c)}. \tag{4}$$

The inequality (3) ensures that all the communications which use the links $src(d)$ followed by $dst(d)$ will have alternative paths after d is removed. The removal of a dependency d impacts on $\Phi(c)$ as follows:

$$\forall (c, j) \in A(d) \Rightarrow \Phi(c) = \Phi(c) \setminus \{\phi_j(c)\}.$$

3.3 Discussion

It is easy to show that our heuristic results in an optimum adaptivity when there is a single cycle in *ASCDG*. Optimality is not gauranteed in the case of multiple cycles. For a globally optimal solution, we need to consider all the cycles simultaneously.

Restricting the routing functions in various network nodes may also affect reachability of certain communications. The order in which the cycles in *ASCDG* get treated may finally decide if the constraint (3) can be met for all cycles or not. This implies that if we look at cycles in one order only then we may not get a routing path for some communications. In fact, in the worst case, we may have to exhaustively consider all possible combinations of dependencies, one from each cycle in *ASCDG*, to be removed to find a feasible minimal routing for all communicating pairs.

3.4 Routing Tables

For each node $p \in P$, and for each input channel $l \in L_{in}(p)$ there is a routing table $RT(p, l)$ in which each entry consists of 1) a *destination address* $d \in P$ and 2) a set of output channels $O \in \wp(L_{out}(p))$ that can be used to forward a message received from channel l and destined to node d. Formally

$$RT(p, l) = \{(d, O) |\ d \in P,\ O = R(p)(l, d) \wedge O \neq \emptyset\}.$$

The routing table of a node $p \in P$ is the union of routing tables of each input channel of p:

$$RT(p) = \bigcup_{l \in L_{in}(p)} RT(p,l).$$

The size of the routing tables depends on both the size of the NoC and the communication density (i.e., the ratio between the number of communications and the number of tasks). Area of a NoC router required by APSRA based algorithm is expected to be larger than the router for the other algorithms. This is because APSRA requires tables (memory) within each router to store routing information. The table based implementation of an APSRA algorithm could also be a blessing because it allows configurability (and even dynamic re-configurability) of the routing algorithm to allow modifications in communication requirements in the running applications. However, routing table size can be reduced by using, for instance, the approach proposed in [12].

4. ADAPTIVITY ANALYSIS

Adaptiveness is an important factor for message routing [7]. Adaptiveness increases the chances that packets may avoid hot spots or faulty components and reduces the risks that packets are continuously blocked. In this section we evaluate the adaptiveness provided by APSRA by using three different communication scenarios: two synthetic, and one that models a real multimedia application.

In both synthetic communication graphs the number of nodes (tasks) is fixed whereas the number of edges (communications) is a parameter that characterise the communication scenario. We define the *communication density* ρ as the ratio between the number of communications and the number of tasks. The synthetic communication graphs are generated randomly based on two different assumptions. In the first synthetic communication graph each task can communicate with every other task with equal probability. In the second one, tasks communicate with a probability depending on the distance between the nodes where they are mapped on. More precisely we define the *one-hop probability ohp* as the probability that a task t_s can communicate with another task t_d when the minimum number of hops from $M(t_s)$ to $M(t_d)$ is equal to 1. The communication probability from t_s to t_d such that the minimum number of hops from $M(t_s)$ to $M(t_d)$ is equal to h is given by:

$$CP(h) = \frac{1}{2}\left[1 - \sum_{i=1}^{h-1} CP(i)\right],\ h \geq 2,$$

with $CP(1) = ohp$. The mapping function is defined as $M(t_i) = p_{i \bmod |P|}$ where $|P|$ is the number of network nodes.

We compare APSRA with adaptive routing algorithms based on the *turn model* [7] and with the *Odd-Even* turn model [4]. They are also the most cited adaptive routing algorithms proposed for mesh networks without using virtual channels. Recent algorithms such as DyAD [9] and contention-look-ahead routing algorithm [14] have not been considered for the following reasons. DyAD is a methodology which can be applied to any adaptive routing algorithm including APSRA. The second one has not been proved to be deadlock free which is a necessary condition in our work. Use of virtual channels will also improve network performance for algorithms produced by APSRA.

Figure 2(a) shows the average degree of adaptiveness for different NoC size and for $\rho = 2$. Each point of the graph has been obtained evaluating 100 random communication graphs and reporting the mean value and the 90% confidence interval. The algorithms based on turn model outperform APSRAs in degree of adaptiveness for sizes of mesh larger than 9. Unfortunately the degree of adaptiveness provided by the turn model is highly uneven [4] (we used the standard deviation of the degree of adaptiveness for the communicating pairs as degree of eveness measure). This is because at least half of the source-destination pairs are restricted to having only one minimal path, while full adaptiveness is provided for the rest of the pairs. On the other side, Odd-Even is the worst one in terms of average degree of adaptiveness but it is more even for different source-destination pairs. APSRA outperforms the other algorithms for small NoC size, but performance decrease very fast as NoC size increases and communication density increase. At any rate this traffic scenario is not very representative for a NoC system. Usually, in fact, cores that communicate most are mapped close to each other [10, 11, 1]. Figure 2(b) shows results obtained for $ohp = 0.4$. In this case APSRA outperforms the other algorithms both in terms of adaptiveness and eveness. APSRA provides 10% and 18% higher adaptivity as compared to turn model based algorithms and Odd-Even algorithm respectively.

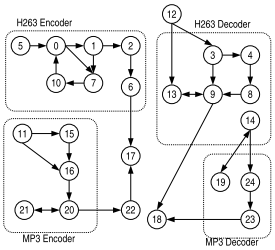


Figure 3: Communication graph of the multimedia system.

As a more realistic communication scenario we consider a generic MultiMedia System which includes an h263 video encoder, an h263 video decoder, an mp3 audio encoder and an mp3 audio decoder [10] whose communication graph is depicted in Figure 3. The application is partitioned into 40 distinct tasks and then these tasks were assigned and scheduled onto 25 selected IPs. The topological mapping of IPs into tiles of a 5×5 mesh-based NoC architecture has been obtained by using the approach presented in [1]. The routing algorithm generated by APSRA is fully adaptive for this specific application whereas algorithms based on turn model and Odd-Even have an average degree of adaptiveness of 0.93 and 0.90 respectively.

5. PERFORMANCE EVALUATION

In this section we evaluate performance of APSRA using a flit-accurate simulator developed in SystemC. As performance metrics we choose *throughput* and *delay*. Comparison is carried out on both synthetic and real traffic scenarios. The evaluations were made on a 8×8 network using wormhole switching with a packet size randomly distributed between 2 and 16 flits. In our model, each router has an input-buffer size of 2 flits. The maximum bandwidth of each link is set to 1 flit per cycle. We use the source packet generation rate as load parameter with Poisson packet injection distribution. For each load value, latency values are averaged over 60,000 packet arrivals after a warm-up session of 30,000 arrived packets. The 95 percent confidence intervals are mostly within 2 percent of the means. For adaptive routing algorithms, we use two different selection policies: *random* and *buffer level*. If multiple output ports are available for a header flit, a random selection policy selects the output randomly whereas the buffer level policy selects the output whose connected input port has the minimum buffer occupied.

Figure 4 shows delay variation and throughput variation for *Trans-*

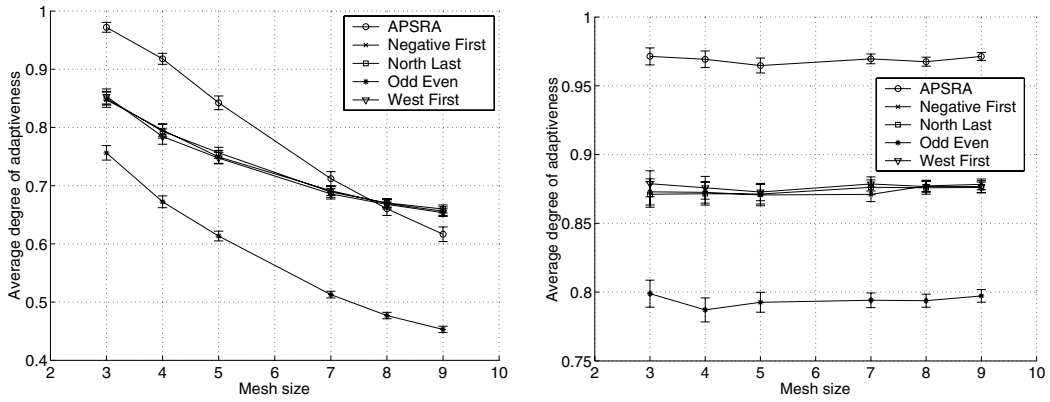


Figure 2: Average degree of adaptiveness for different NoC size for random generated communication graph with $\rho = 2$ (a) and locality traffic with $\rho = 2$ and $ohp = 0.4$.

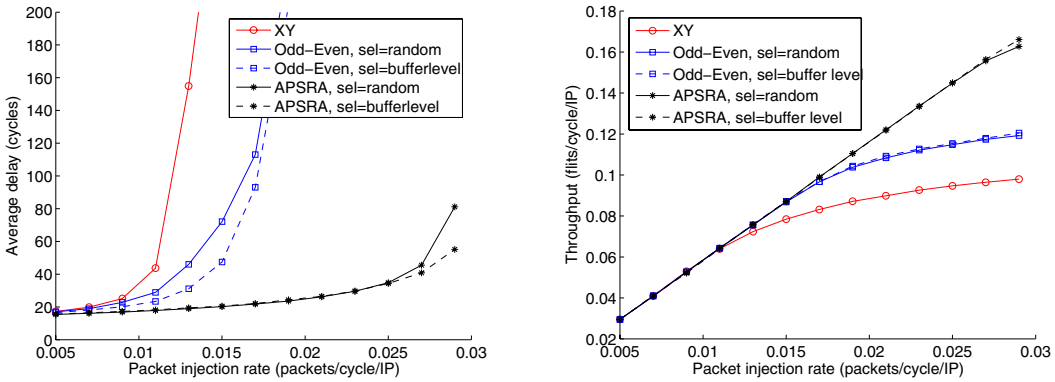


Figure 4: Delay variation (a) and throughput variation (b) under *Transpose 1* traffic.

Table 1: Improvement in average delay of APSRA as compared to XY and Odd-Even for different traffic scenarios.

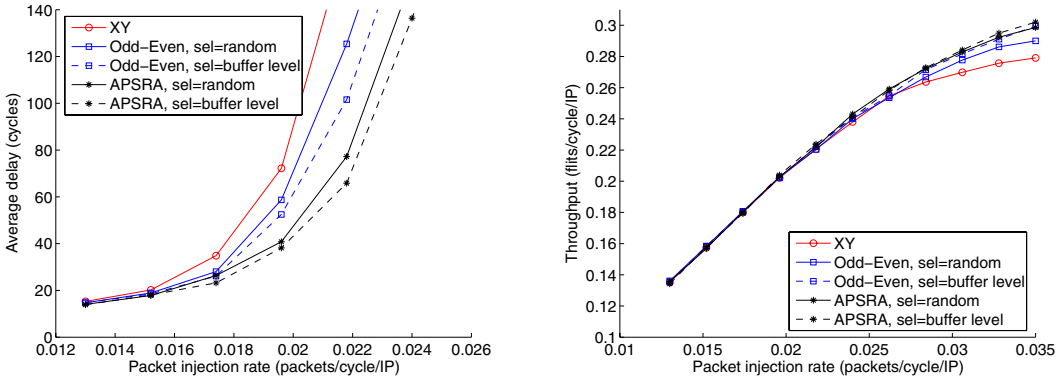
Traffic scenario	pir (pkt/cyc/IP)	Avg. delay (cycles) XY	OE	APSRA	APSRA impr. vs. XY	vs. OE
Random	0.010	68	51	34	49.8%	32.8%
Locality	0.020	39	34	29	24.2%	13.2%
Transpose 1	0.011	91	39	19	79.2%	51.1%
Transpose 2	0.011	82	31	19	76.6%	38.4%
Hotspot-4c	0.003	46	50	34	26.7%	32.1%
Hotspot-4tr	0.003	52	37	30	42.2%	17.5%
Hotspot-8rs	0.003	34	25	20	41.8%	21.5%
Mms	0.020	36	30	23	36.1%	23.3%
Average improvement					47.1%	28.7%

pose 1 traffic scenario [7]. In *Transpose 1* traffic a node (i, j) only sends messages to a node $(7-i, 7-j)$. As we can see, APSRA outperforms the other routing strategies. Maximum load before the network starts saturating[1] using a deterministic routing is 0.012 packets/cycle/IP. This value increases to 0.017 packets/cycle/IP when Odd-Even is used and reaches is 0.028 packets/cycle/IP with APSRA.

[1]A network is said to start saturating when increase in applied load does not result in linear increase of throughput.

For space reason, the results obtained for other traffic scenarios are summarized in Table 1. For each traffic scenario and for each routing algorithm the table reports the average delay measured at an injection load below saturation (column *pir*). *Random* traffic refers to the uniform random traffic in which a node send a message to any other node with equal probability. *Locality* traffic has been generated with $ohp = 0.4$. In *Transpose 2* traffic a node (i, j) only sends messages to node (j, i). In the hot spot scenario, some nodes are designated as the *hot spot nodes*, which receive hot spot traffic in addition to regular uniform traffic. Given a hot spot percentage h, a newly generated packet is directed to each hot spot node with an additional h percent probability. We consider three hot spot traffic scenarios. In *Hotspot-4c* the hot spot nodes are located at the center of the mesh ((3,3), (4,3), (3,4), (4,4)) with 20% hot spot traffic. In *Hotspot-4tr* the hot spot nodes are located at the top-right corner of the mesh ((0,6), (0,7), (1,6), (1,7)) with 20% hot spot traffic. Finally, in the *Hotspot-8rs* the hot spot nodes are located at the right side of the mesh $((i,7), i = 0,1,\ldots,7)$ with 10% hot spot traffic.

Finally, as a real communication scenario we use the Multimedia System (Mms) described in [10] whose communication graph is depicted in Figure 3. For this scenario we consider self-similar packet injection distributions. Self-similar traffic has been observed in the bursty traffic between on-chip modules in typical MPEG-2 video applications [13] and networking applications. Figure 5 shows the



(a) (b)

Figure 5: Delay variation (a) and throughput variation (b) under traffic generated by a multimedia system.

average delay and throughput variation for different injection loads. For an injection load of 0.020 packets/cycle/IP, that is below the saturation for all the routing algorithms, the average delay is 36 cycles for XY, 32 cycles for Odd-Even, and 26 for APSRA when a random selection policy is used. By using a selection policy based on buffer levels, average delay decreases to 30 cycles for Odd-Even and to 23 cycles for APSRA.

In conclusion, on average, APSRA outperforms both deterministic XY and adaptive Odd-Even by almost 50% and 30% respectively in terms of average delay.

6. CONCLUSIONS

In this paper, we have made a case for application specific routing in NoC systems and proposed a methodology to design such routing algorithms. Our methodology is general and can be applied to design application specific deadlock free routing algorithms for any topology. We have shown that, for homogeneous NoC architecture with 2-dimensional topology, algorithms designed by our methodology offer higher adaptivity and higher performance as compared to the general purpose routing algorithms. Our methodology can be used to generate deadlock free routing algorithms for non-homogeneous mesh topology NoC incorporating concept of regions. However, higher performance comes at the cost of router tables. There are aspects of application specific communication other than communication topology which can be exploited to further increase the communication performance in NoC systems. Information about traffic classes and the information about communication schedule are definite candidates for this purpose.

7. ADDITIONAL AUTHORS

Additional author: Vincenzo Catania (DIIT, University of Catania, Italy, email: `vcatania@diit.unict.it`).

8. REFERENCES

[1] G. Ascia, V. Catania, and M. Palesi. Multi-objective mapping for mesh-based NoC architectures. In *Second IEEE/ACM/IFIP Int. Conf. on Hardware/Software Codesign and System Synthesis*, pages 182–187, Stockholm, Sweden, Sept. 8–10 2004.

[2] A. Bartic, J.-Y. Mignolet, . Nollet, T. Marescaux, D. Verkest, S. Vernalde, and R. Lauwereins. Highly scalable network on chip for reconfigurable systems systems. In *Int. Conf. on System-On-Chip*, pages 79–82, Tampere, Nov. 2003.

[3] E. Bolotin, A. Morgenshtein, I. Cidon, and A. Kolodny. Automatic and hardware-efficient SoC integration by QoS network on chip. In *IEEE Int. Conf. on Electronics, Circuits and Systems*, Tel Aviv, Dec. 2004.

[4] G.-M. Chiu. The odd-even turn model for adaptive routing. *IEEE Trans. on Parallel Distribuited Systems*, 11(7):729–738, 2000.

[5] W. J. Dally and B. Towles. Route packets, not wires: On-chip interconnection networks. In *Design Automation Conference*, pages 684–689, Las Vegas, Nevada, USA, 2001.

[6] J. Duato. A new theory of deadlock-free adaptive routing in wormhole networks. *IEEE Trans. on Parallel and Distribuited Systems*, 4(12):1320–1331, Dec. 1993.

[7] C. J. Glass and L. M. Ni. The turn model for adaptive routing. *Journal of the Association for Computing Machinery*, 41(5):874–902, Sept. 1994.

[8] R. Holsmark and S. Kumar. Design issues and performance evaluation of mesh NoC with regions. In *IEEE Norchip*, pages 40–43, Oulu, Finland, Nov.21–22 2005.

[9] J. Hu and R. Marculescu. DyAD - smart routing for networks-on-chip. In *ACM/IEEE Design Automation Conference*, pages 260–263, San Diego, CA, USA, June 7–11 2004.

[10] J. Hu and R. Marculescu. Energy- and performance-aware mapping for regular NoC architectures. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 24(4):551–562, Apr. 2005.

[11] S. Murali and G. D. Micheli. Bandwidth-constrained mapping of cores onto NoC architectures. In *Design, Automation, and Test in Europe*, pages 896–901. IEEE Computer Society, Feb. 16–20 2004.

[12] M. Palesi, S. Kumar, and R. Holsmark. A method for router table compression for application specific routing in mesh topology NoC architectures. In *SAMOS VI Workshop: Embedded Computer Systems: Architectures, Modeling, and Simulation*, pages 373–384, Samos, Greece, July 17–20 2006.

[13] G. Varatkar and R. Marculescu. Traffic analysis for on-chip networks design of multimedia applications. In *ACM/IEEE Design Automation Conference*, pages 510–517, June 2002.

[14] T. T. Ye, L. Benini, and G. D. Micheli. Packetization and routing analysis of on-chip multiprocessor networks. *Journal of System Architectures*, 50(2-3):81–104, 2004.