APSRA: A methodology for design of Application Specific Routing Algorithms for NoC Systems*

Maurizio Palesi
Vincenzo Catania
Univeristà di Catania, Italy

Rickard Holsmark
Shashi Kumar
Jönköping University, Sweden

April 6, 2006

Abstract

A future NoC architecture must be general enough to allow volume production and must have features to specialize and configure to match and meet application's performance requirements. In this report, we present a methodology to specialize the routing algorithm in NoC routers to optimize its communication performance while ensuring deadlock free routing. Duato's theory of deadlock free routing is extended to incorporate application's communication requirements to improve routing adaptiveness. We demonstrate through analysis and modeling and evaluation that routing algorithms produced by our methodology have higher adaptiveness and higher performance as compared to general purpose deadlock free routing algorithms.

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

1 Introduction

Advances in technology now make it possible to integrate hundreds of cores (e.g. general or special purpose processors, embedded memories, application specific components, mixed-signal I/O cores) in a single silicon die. The large number of resources that have to communicate makes the use of interconnection systems

*This document is available from the Dipartimento di Ingegneria Informatica e delle Telecomunicazioni at the Università degli Studi di Catania, V.le Andrea Doria 6—I95125 Catania, Italy, as technical report DIIT-TR-01-060406, March 2006. Please, use the technical report number when you reference this document. Authors' addresses: M. Palesi and V. Catania, Dipartimento di Ingegneria Informatica e delle Telecomunicazioni, V.le Andrea Doria, 6, 95125 Catania, Italy, {mpalesi,vcatania}@diit.unict.it; R. Holsmark and S. Kumar, School of Engineering,J¨onk¨oping University Box 1026, SE-55111 J¨onk¨oping, Sweden, {Rickard.Holsmark,Shashi.Kumar}@ing.hj.se.

based on shared buses inefficient. One way to solve the problem of on-chip communications is to use a Network-on-Chip (NoC)-based communication infrastructure. These architectures emphasize the separation between computing and communication, and guarantee a good degree of design reuse and scalability.

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 [10, 16, 5, 15, 18] 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.

It is now possible to envision a scenerio in which a mesh topology NoC chip, populated with an application area specific set of cores, will be available as off the self 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. A good adaptive routing algorithm should have both low average latency for messages and freedom from 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 [7]. 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, 11]. 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] have proposed hard coded paths for deadlock safe routing for an application. In their approach, the possibility of deadlock for the application communication scenario is analyzed and solved off-line. Any change in traffic patterns results in a complete re-analysis of deadlock freeness and may result in changes to be made to affected paths. A non-minimal deadlock free routing algorithm is described for a irregular mesh topology NoC with regions in [11]. This algorithm is biased in favor of some area of the network as compared to the other area.

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 dead-

lock. 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 report, 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 analyzed the generated algorithms for a large number of synthetic communication graphs as well as a graph corresponding to a real application. We show through analytical analysis that generated algorithms have significantly higher adaptivity as compared to the well known deadlock free routing algorithms especially when the communication traffic has neighborhood behavior. We have also evaluated and compared the performance of APSRAs with Odd-Even algorithm through modeling and simulation. Again, we observe that the average latency of routing algorithms generated through our methodology is smaller for low traffic load.

The report is organized as follows. Section 2 provides the terminology, a set of definitions and the theorem at the heart of the proposed methodology. Section 3 presents the APSRA design methodology. Adaptivity analysis and comparison with current deadlock free adaptive routing algorithms for different traffic scenarios is presented in Section 4. Section 5 reports dynamic performance evaluation results for both synthetic and real traffic scenarios. Finally, Section 6 concludes the report and outlines some directions for future work.

2 Terminology and Definitions

In this section we define the concept of *Application Specific Routing*. In an embedded system scenario the communication traffic between the 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. This information can also be incorporated in Duato's theory for systematic design of deadlock free routing algorithms for communication networks [6].

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 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 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 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 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 $q \in P$. We assume that $R(p)(l,q) = \emptyset$ is q is not reachable from p.

The $\wp$ indicates a power set. We indicate with R the set of all routing functions:

$$R = \{R(p) :\ p \in P\}.$$

Definition 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

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

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 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 2.1 *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 the 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* and 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 2.1 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 remove an application specific dependency (i.e. break a cycle of 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 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:

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

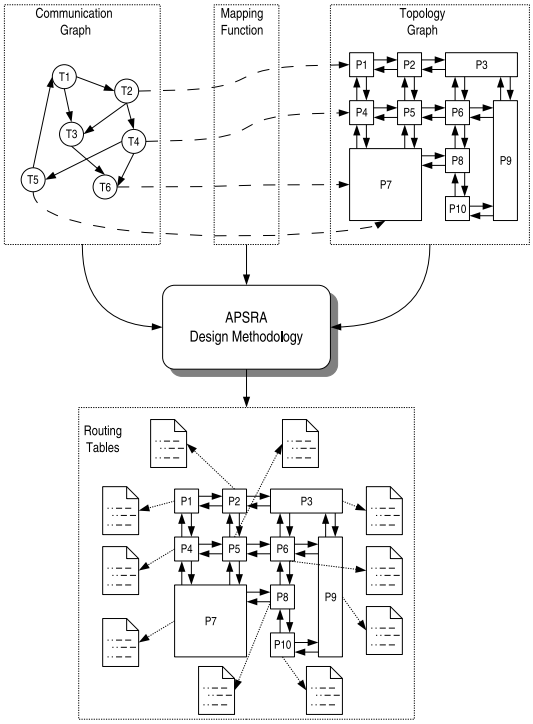


Figure 1: Overview of the APSRA design methodology.

Given a communication $c \in C$ we indicate with $\Phi(c)$ the set of all 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

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

Theorem 3.1 *Given an ASCDG*(L,D) *and* $d = (l_i, l_j) \in D$ *then* $A(d) \neq \emptyset$.

Proof $\Rightarrow d = (l_i, l_j) \in D$ then $\exists\ c \in C$ such that $l_j \in R(dst(l_i))(l_i, c)$ that is, there exists a communication c which has a path that contains both l_i and l_j. This path belongs to $\Phi(c)$ and suppose it is the j-th path of $\Phi(c)$ named $\phi_j(c)$. Then the pair (c, j) belongs to $A(d)$ because $src(d) = l_i \in \phi_j(c)$ and $dst(d) = l_j \in \phi_j(c)$.

$\Leftarrow$ Let $a = (c, j) \in A(d)$, then $\exists \phi_j(c) \in \Phi(c)$ that contains both $src(d) = l_i$ and $dst(d) = l_j$. The condition (1) is satisfied by construction because $d \in D$. The existence of the path $\phi_j(c)$ states that a communication c traveling on l_i can immediately use l_j. This means that the routing function at node $dst(l_i)$ allows this turn. Therefore $l_i \in R(dst(l_i))(l_i, M(dst(c)))$ and the condition (2) is satisfied too.

Definition 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 minimum 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 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 minimum 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 3.3) 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 adptiveness 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 as a candidate dependence to be removed the $d \in D_c$ 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:

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

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 mimimal routing for all communicating pairs.

3.3 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 betwen the number of communications and the number of tasks).

4 Adaptivity Analysis and Comparison

In this section we test APSRA methodology 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 betwen 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 of 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:

$$\begin{cases} CP(1) = ohp \\ CP(h) = \frac{1}{2}\left[1 - \sum_{i=1}^{h-1} CP(i)\right]. \end{cases}$$

The mapping function is defined as $M(t_i) = p_{i\%|P|}$ where the symbol % is the module operator and $|P|$ is the number of network nodes.

We compare APSRA to adaptive routing algorithms based on the *turn model* [9] and to the *Odd-Even* turn model [4]. We first analyse the different algorithms in terms of degree of adaptiveness. Figure 2 shows the average degree of adaptiveness and the standard deviation of degree of adaptiveness for different NoC size and for two communication densities ($\rho = 2$ and $\rho = 4$). Each point of the graph has been obtained evaluating 100 random communication graphs and reporting the

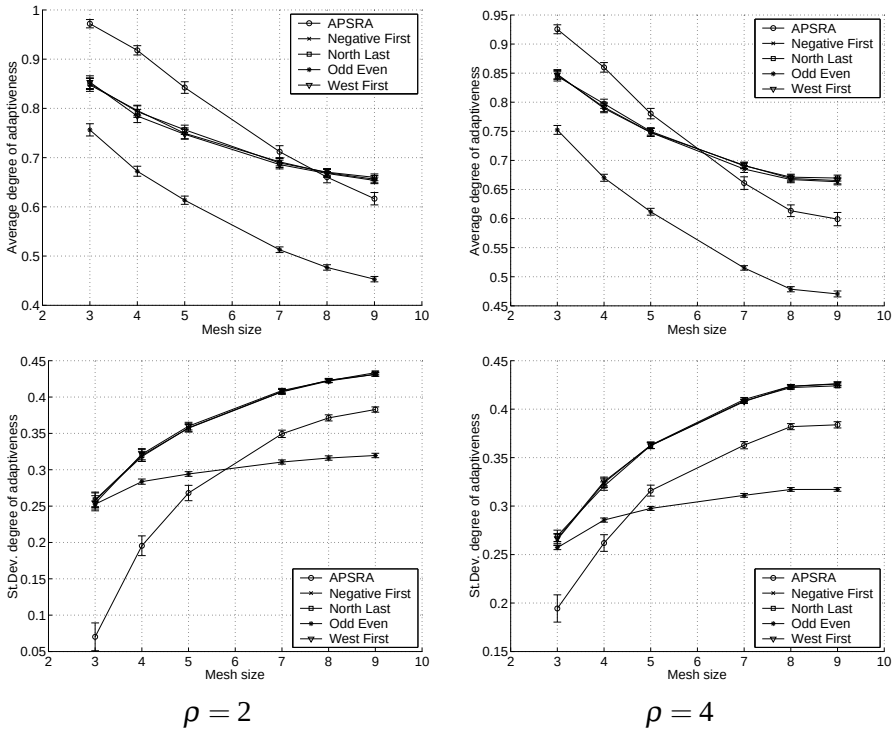


Figure 2: Average degree of adaptiveness and standard deviation for different NoC size and for random generated communication graph with two different communication densities.

mean value and the 90% confidence interval. As expected the algorithms based on turn model outperform in degree of adaptiveness. Unfortunately the degree of adaptiveness provided by the turn model is highly uneven [4]. 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. This is confirmed by the high standard deviation values these algorithms exhibit [Figure 2(b)]. 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 outperform the other algorithms for small NoC size, but performance decrease very fast as NoC size increase 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 [14, 17, 1].

The second traffic scenario overcome this problem. Figure 3 shows results obtained for $ohp = 0.4$. In this case APSRA outperform the other algorithms both in terms of adaptiveness and standard deviation. Quantitatively, APSRA provide very high level of adaptivity in average over 10% and 18% respectively for turn model based algorithms and Odd-Even for $\rho = 2$, and over 7% and 15% respectively for turn model based algorithms and Odd-Even for $\rho = 4$. Moreover, the degree

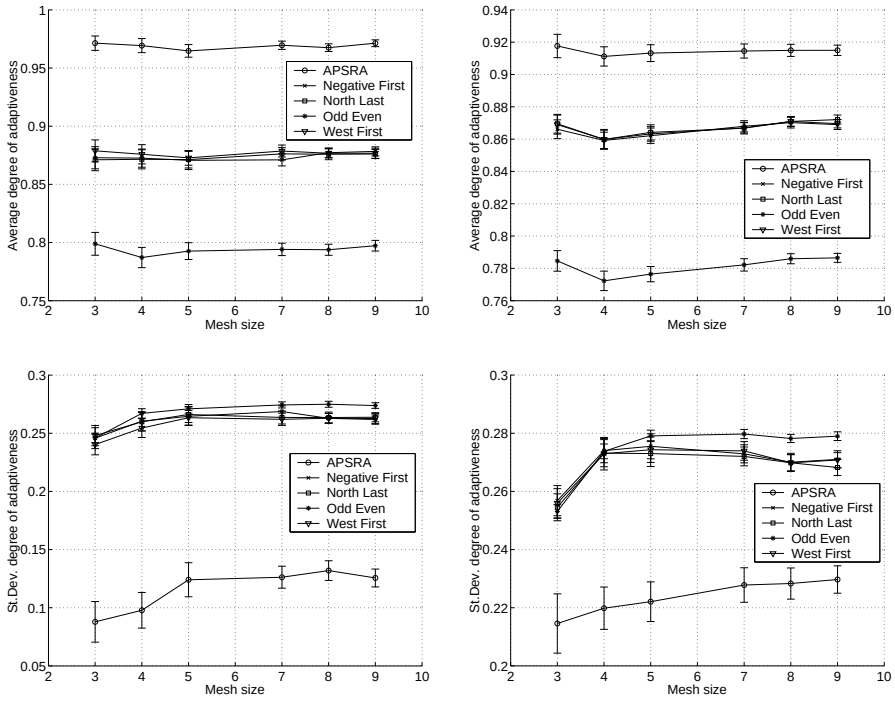


Figure 3: Average degree of adaptiveness and standard deviation for different NoC size and for random generated communication graph with two different communication densities and with $ohp = 0.4$.

of routing adaptiveness provided by APSRA is more even for different source-destination pairs.

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 [12] (see Figure 4). 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

We also evaluated APSRA using a flit-level simulator developed in SDL [8] to verify if the promising analytical results of adaptiveness also translate to increase in performance. We compare APSRA with ODD Even because the latter has been

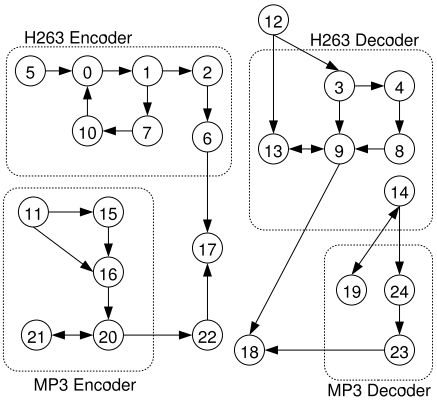
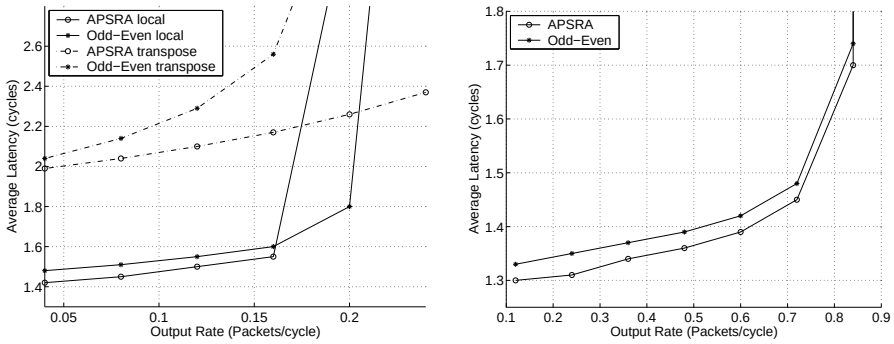


Figure 4: Communication graph of the multimedia system.

proved to exhibit the best performance among different traffic scenarios [4]. The evaluations were made using wormhole switching with a packet size of 10 flits. In our model, each router has an input-buffer size of 2 flits and an output-buffer size of 1 flit. If multiple output ports are available for a header flit, a random selection is made. The maximum bandwidth of each link is set to 1 packet per cycle. We use the source packet generation rate as load parameter. For each load value, latency values are averaged over 60000 packet arrivals after a warm-up session of 30000 arrived packets. We present results on average latency where throughput levels are below saturation. We define latency as the duration, in terms of network cycles, between creation of a packet at the source until the last flit has reached the destination. The delays between packets are varied according to a Poisson distribution. First we show results from transpose and random with locality generated communication patterns, simulated on an 8×8 network. For random with locality pattern, the latency values are averaged over 10 different mappings. In Figure 5(a), we see that on lightly loaded situations, our algorithm gives a decrease in latency of about 4% in the random with locality set-up. However, closer to saturation Odd-Even algorithm performs better. For the communication traffic corresponding to transpose pattern APSRA has a significantly higher performance. Here the advantage in latency also grows with increased load.

We also made simulations on the application specific scenario, described in [12]. We set the packet generation rate different at the sources, corresponding to the data to be transferred to each destination node. The data rates are then equally scaled up to see the effect of increased load. In this situation, shown in Figure 5(b), APSRA has an advantage of lower latency, but the saturation point occurs at similar load levels. Note, that the output rate corresponds to the source with the highest rate, as this is a bottleneck in this case.

APSRA clearly has an advantage in terms of lower latency on lightly loaded networks, due to its higher adaptivity. However, when networks become congested and randomness is used in traffic patterns, the adaptivity no longer pays off. This is



(a) (b)

Figure 5: Average latency vs. load for transpose and random traffic with locality (a), and for specific traffic (b).

a phenomena also documented in earlier research [4, 13, 19]. We believe though, that such situations are not likely to be the case for real NoC.

6 Conclusions

In this report, 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. We plan to use our methodology to generate deadlock free routing algorithms for non-homogeneous mesh topology NoC incorporating concept of regions and compare their performance with other proposed solutions. However, higher performance comes at the cost of larger router tables. We are currently working on techniques for loss-less compression of these routing 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.

References

[1] G. Ascia, V. Catania, and M. Palesi. Multi-objective mapping for mesh-based NoC architectures. In *Second IEEE/ACM/IFIP International Conference 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 *International Conference 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 International Conference on Electronics, Circuits and Systems*, Tel Aviv, Dec. 2004.

[4] G.-M. Chiu. The odd-even turn model for adaptive routing. *IEEE Transactions 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 Transactions on Parallel and Distribuited Systems*, 4(12):1320–1331, Dec. 1993.

[7] J. Duato, S. Yalamanchili, and L. Ni. *Interconnection Networks: An Engineering Approach*. Morgan Kaufmann, 2002.

[8] J. Ellsberger, D. Hogrefe, and A. Sarma. *SDL Formal Object-oriented Language for Communicating Systems*. Prentice Hall, 1997.

[9] 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.

[10] P. Guerrier and A. Greiner. A generic architecture for on-chip packet-switched interconnections. In *Design Automation and Test in Europe*, pages 250–256, Paris, France, 2000.

[11] 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.

[12] J. Hu and R. Marculescu. Energy-aware mapping for tile-based NoC architectures under performance constraints. In *Asia & South Pacific Design Automation Conference*, pages 233–239, Jan. 2003.

[13] 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.

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

[15] F. Karim, A. Nguyen, and S. Dey. An interconnect architecture for networking systems on chips. *IEEE Micro*, 22(5):36–45, Sept.–Oct. 2002.

[16] S. Kumar, A. Jantsch, J.-P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, and A. Hemani. A network on chip architecture and design methodolog. In *IEEE Computer Society Annual Symposium on VLSI*, page 117, 2002.

[17] 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.

[18] P. P. Pande, C. Grecu, A. Ivanov, and R. Saleh. Design of a switch for network on chip applications. In *IEEE International Symposium on Circuits and Systems*, volume V, pages 217–220, Bangkok, Thailand, May 2003.

[19] P. P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh. Performance evaluation and design trade-offs for network-on-chip interconnect architectures. *IEEE Transactions on Computers*, 54(8):1025–1040, Aug. 2005.