Exploiting Communication Concurrency for Efficient Deadlock Free Routing in Reconfigurable NoC Platforms

Maurizio Palesi[1], Shashi Kumar[2], Rickard Holsmark[2], and Vincenzo Catania[1]

[1]Dept. of Computer Science and Telecommunication Engineering
University of Catania, Italy
{mpalesi, vcatania}diit.unict.it

[2]Embedded Systems Department
Electronics and Computer Engineering
School of Engineering, Jönköping University, Sweden
{Shashi.Kumar, Rickard.Holsmark}@jth.hj.se

Abstract

In this paper we make a case for the use of NoC paradigm to develop future FPGAs in which large computational blocks (cores) are connected to each other through a packet switched communication network. We propose a methodology to develop efficient and deadlock free routing algorithms for such NoC platforms which can be specialized for an application or a set of concurrent applications. Application specific topology of communicating cores as well as information about their communication concurrency over time is exploited to maximize communication adaptivity and performance. We demonstrate, both through analysis of adaptivity as well as simulation based evaluation of latency and throughput, that our algorithm gives significantly higher performance as compared to general purpose deadlock free algorithms like XY and Odd-Even.

1. Introduction

There is a general consensus that Network on Chip (NoC) paradigm is one of the important alternatives for implementing multi-core heterogeneous Systems on Chip (SoCs). Still only a few publications report the actual use of NoC idea for implementing real application specific SoCs. Low volume of production continues to be an argument against large investments in a relatively untested technology. Of course there are other reasons, including non-availability of development tools, which prevent use of NoC idea in design of industrial SoCs. One way to practically afford large development cost of a NoC system is to amortize it among a large number of applications.

It is possible to envision a chip based on NoC paradigm as a future FPGA. Such Field Programmable Resource Array (FPRA) will use grosser level of configurable computational resources and grosser level of configurable packet switched communication resources. One can easily envision a scenario in which a mesh topology NoC chip populated with an application area specific set of cores could be available as off the shelf standard product. Such future FPGAs are likely to be provided with small table based routers for efficient, adaptive and deadlock free routing among cores. This paper is in the direction of development of routers for such application area specific field programmable NoC systems.

One of the main criticism about adaptive routing algorithms is related to the problem that packets can reach the destination in an out-of-order fashion due to the difference in congestion levels on the multiple paths. However, this problem can be efficiently coped with by using the simple re-ordering mechanism at network re-convergent nodes proposed in [16]. Adaptive routing algorithms, if not designed carefully, have a danger of causing traffic deadlocks. Many deadlock free routing algorithms have been proposed in literature for mesh topology networks [4, 8, 12]. In most of these algorithms freedom from deadlock is achieved at a high loss of adaptivity. Odd-Even routing algorithm [4] provides deadlock free routing only for homogeneous mesh topology NoC architectures. Bolotin *et al.* [2] have proposed static hard coded paths for deadlock safe routing for an application in a heterogeneous mesh topology NoC. A non-minimal deadlock free routing algorithm is described for an irregular mesh topology NoC with regions in [9]. Based on the concept of channel dependency graphs given by Dally and Seitz [5], 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 takes only the network topology as input and generates a routing algorithm which will work for all possible communication traffic situations.

1-4244-0910-1/07/$20.00

A NoC system which is specialized for a specific application or a set of concurrent applications can be considered as a semi-static system. We have the information about the set of pairs of cores which communicate and other pairs which never communicate. After application mapping and scheduling phase of system development, we also have information about the set of communication transactions which are concurrent and others which are non-concurrent.

Such information has been taken into account by Murali *et al.* in [17, 18]. In [17] the authors motivate the need to consider multiple use-cases during the NoC design process and present a method to map the applications onto the NoC architecture, satisfying the design constraints of each-individual use-case. The proposed method, which works on a synthetic worst-case use-case, performs poorly on systems where the traffic characteristics of the use-cases are very different or when the number of use-cases are large. The same authors, in [18], present a variant of the method that resolve such problems. However, both the methods, do not take advantage of the knowledge of the use-cases to improve the underlying routing algorithm. They, in fact, address only the mapping problem and consider a fixed static path routing as implemented in the *Æthereal* architecture [21]. The problem of switching between different use-cases requires the presence of mechanisms to manage the dynamic reconfiguration of the network assuring, for instance, deadlock freeness during the transition periods. This problem has been addressed by Duato *et al.* in [7] and by Lysne *et al.* in [15] and will be referenced again later in this paper.

In [19] we extended Duato's theory and presented a methodology to exploit static communication topology of an application to develop efficient deadlock free routing algorithms for NoC systems. In this paper, we extend the methodology presented in [19] to exploit information about an application's dynamic communication topology (concurrency of communication transactions) for the same purpose. We show that this extension leads to a significant improvement of performance of the resulting routing algorithm. We think these results can be used for development of future field programmable NoC systems.

2. Terminology and Definitions

In this section we briefly report some definitions which will be used in the rest of the paper.

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 $p \in P$.*

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 $q \in P$.*

Where $L_{in}(p)$ and $L_{out}(p)$ are the set of input channels and output channels for node 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*

$$dst(l_i) = src(l_j) \tag{1}$$

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

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 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.*

In [19] we have proved the following theorem.

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.*

3. APSRA Design Methodology

A brief overview of the APSRA design methodology [19] is depicted in Figure 1. The input to the methodology is the application modeled as a task graph along with the NoC architecture in which the table based routers can be configured. We assume that the tasks in the application have already been mapped and scheduled on the available NoC

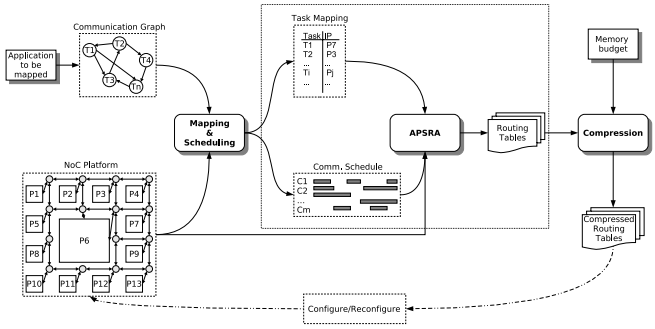


Figure 1. Overview of the APSRA design methodology.

resources. Using this information, APSRA generates a set of routing tables (one for each router of the NoC) which not only guarantee both the reachability and the deadlock freeness of communication among tasks but also try to maximise routing adaptivity [4, 8]. Adaptivity ((a.k.a. degree of adaptiveness) is an important metric for judging the ability of a network to provide alternative paths to packets to handle contention and faults. A network with high adaptivity helps packets to avoid hot spots or faulty components and reduces the chances that packets are continuously blocked. The information about communication concurrency could also be exploited to improve routing adaptivity and represents the main contribution of this paper. Finally, a compression technique can be used to compress the generated routing tables [20].

There are many ways in which such a NoC architecture can evolve. For a custom NoC solution, a set of IP cores are selected matching application characteristics. Tasks are then mapped and scheduled on these IP cores before these cores are assigned slots in the given network topology of the architecture. On the other extreme the implementor may get an application area specific general purpose configurable FPRA chip in which IP cores are already placed in specific positions in the network. Then a tool is required for mapping tasks to the available IP cores and then scheduling their computation on the resources and scheduling their communication using the network. These problems related to NoC architecture design and mapping applications to a fixed NoC architecture have been extensively addressed in literature [1, 3, 13, 14]. In this paper we refer to the latter scenario.

For the sake of example, let us consider the communication graph and the topology graph depicted in Figure 2(a) and 2(b) respectively. Although for this example the topology is mesh-based, the approach is general and can be applied to any network topology without any modification. As mapping function let us consider $M(T_i) = P_i, i = 1,2,\ldots,6$. The channel dependency graph (*CDG*) [6] for a minimal fully adaptive routing algorithm is shown in Figure 2(c). Since it contains six cycles, Duato's theorem [6] cannot assure deadlock freeness of the minimal fully adaptive routing for this topology. The number of cycles is reduced to two for the *ASCDG* as shown in Figure 2(d). We observe that some dependencies in the *CDG* are not present in the *ASCDG*. For instance, the edge corresponding to dependency $l_{1,2} \rightarrow l_{2,3}$ in *CDG* does not appear in *ASCDG*. In fact, channels $l_{1,2}$ and $l_{2,3}$ can be used in sequence only for the communications $T_1 \rightarrow T_3$, $T_1 \rightarrow T_6$, and $T_4 \rightarrow T_3$ which are not present in the *CG*. Although in this case we still cannot assure deadlock freeness, we can simply break the cycle as follows. The application specific channel dependency $l_{4,1} \rightarrow l_{1,2}$ is due to the communication $T4 \rightarrow T2$. Such communication can be realized by both paths $P4 \rightarrow P5 \rightarrow P2$ and $P4 \rightarrow P1 \rightarrow P2$. If the routing function is restricted in such a way as the latter path is prohibited, the application specific channel dependency $l_{4,1} \rightarrow l_{1,2}$ does not exist any longer. In a similar way it is possible to break the second cycle, removing, for instance, the dependency $l_{1,4} \rightarrow l_{4,5}$ due to communication $T1 \rightarrow T5$.

However, this restriction reduces the adaptivity of the routing. Now suppose that we have some knowledge about communication concurrency and suppose that communication $T1 \rightarrow T5$ and communication $T2 \rightarrow T4$ do not overlap in time. Figure 2(e) highlight the dependencies due to such communications. Since these communications are not concurrent, the associated dependencies are also not concurrently active. The result is that the two cycles are actually false cycles. In conclusion, for this latter case a minimal fully adaptive routing is deadlock free.

3.1. APSRA Overview

The APSRA methodology can be summarised as follows.

```
APSRA(in: CG, TG, M; out: RT) {
  R ← MinFullyAdaptiveRouting(CG, TG, M);
  BuildASCDG(CG, TG, M, R, ASCDG);
  GetCycles(ASCDG; C);
  RemCycles(C, ASCDG, CG, TG, M, R, success);
  if (success)
     ExtractRoutingTables(R, RT);
  else
     RT ← ∅;
}
```

The `APSRA` procedure gets as inputs a communication graph *CG*, a topology graph *TG* and a mapping function *M* and returns a set of routing tables. The procedure starts initializing *R* with a minimal fully adaptive routing, then calls the procedure `BuildASCDG` which builds the *ASCDG*. The procedure `GetCycles` returns the set *C* of cycles in the *ASCDG*. `RemCycles` tries to remove all the cycles *C*

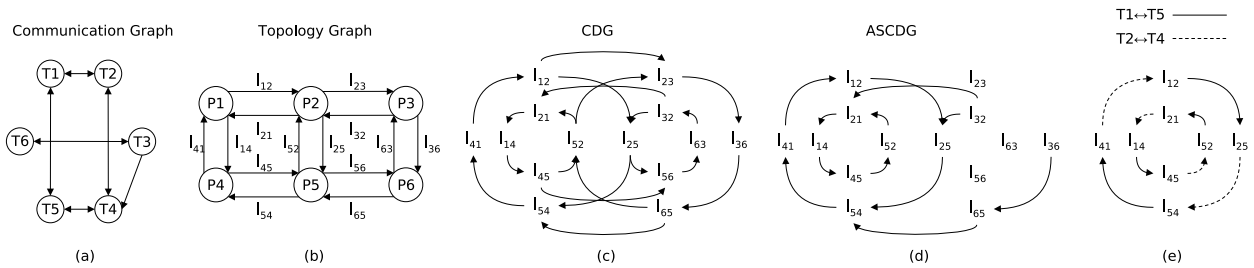


Figure 2. Comparison of cyclic dependencies without and with APSRA methodology.

from *ASCDG* with the objective to minimising the loss in adaptivity and with the constraint to guarantee the reachability between all communicating pairs. If it succeeds, it returns *true* and the procedure `ExtractRoutingTables` is used to extract routing tables from R, otherwise an empty set is returned. For a detailed description of the last two steps see [19].

3.2. Exploiting Communication Concurrency

Starting from an application (or a set of concurrent applications) it is possible to extract a communication graph CG defining which tasks communicate and which do not communicate. After task mapping and scheduling step of system level design, it is possible to know whether two communications are concurrent or not. In general, it is possible to divide the execution of the application (or a set of concurrent applications) into intervals. A generic interval is characterized by a *Communication Scenario* (CS_i). A CS_i can be seen as a subgraph of the CG and specifies which node pairs in the architecture communicate during that interval. CSs in different intervals may be different but some non-consecutive intervals may have the same CS.

Now, let us suppose to have identified a succession of intervals and the associated communication scenarios $CS_i, i = 1,2,\ldots,N$. The APSRA design methodology can be extended to exploit the communication concurrency information as follows.

```
APSRA_Concurrency(in: CS_i,i=1,...,N, TG, M;
                  out: RTv)
{
  for (i=1 to N) {
    APSRA(CS_i.CG, TG, M, RT);
    if (RT=∅)
      printf("No deadlock free algorithm
              exist for scenario %d", i);
    else
      RTv[i] ← RT;
  }
}
```

The input parameters of `APSRA_Concurrency` procedure are the succession of communication scenarios CS_i, the topology graph TG, and the mapping function M. The output, is the vector of routing tables, RTv, one for each interval (i.e., $RTv[i]$ holds the routing tables associated to CS_i). For each communication scenario CS_i the `APSRA` procedure is invoked and acting on the communication graph associated to CS_i. If `APSRA` fails, it returns an empty set and an error message is shown. Such situations occurs if there exists at least one pair communicating tasks such that we can not find a path for them without causing deadlock. Otherwise, if `APSRA` succeeds, the returned set of routing tables is stored in RTv and the next communication scenario is processed.

It should be pointed out that interval based routing scheme has some overhead and issues regarding interval boundaries. The first one is regarding the "cooling period" required to complete all the communications started in a communication scenario before starting the next communication scenario. The second one is regarding reconfiguration of the routing tables before starting a new communication scenario. In this work, we assume that both these intervals are much smaller as compared to the intervals for the communication scenarios. This assumption is not as restrictive as one might intuitively think. As a practical example, let us consider a digital video camera application with the common recording/playback functions. Since such functions are never used simultaneously, it is simple to envision two communication scenarios one for the recording and the other for the playback. In this case, the hypothesis about the negligible duration of both the cooling period and the reconfiguration time as compared to the intervals for the communication scenarios is valid since the time spent to switch between the two scenarios can be considered as negligible as compared to the time spent in recording and playback.

The process of defining communication scenarios is the responsibility of task scheduling phase of NoC specialization. The task scheduler can either use CS period as a constraint, or it can produce communication scenario periods as output while optimizing some objective function. The granularity of intervals will affect routing adaptivity as well

Table 1. Silicon area occupancy in μm^2 for various blocks of a router.

	XY	OddEven	RT4	RT8
Routing func.	803	1365	17051	36384
Input FIFOs	132891	132891	132891	132891
Crossbar	14041	14041	14041	14041
Arbiter	2007	2008	2009	2010
Total	149745	150307	165995	185329

as routing performance. Large intervals will effectively increase communication density leading to decrease in adaptivity [19], but will have lower reconfiguration overheads. On the other hand, routing algorithms generated by APSRA will have almost 100% adaptivity for very small intervals. The drawback is that performance improvement due to this could be lost due to heavy overheads of frequent reconfiguration. Deciding optimal communication scenario intervals is an interesting future research problem.

3.3. Design Issues

Cost of implementing a table in every router is an argument against APSRA methodology. In [20] we presented a method for router table compression for application specific routing in mesh-based NoC architectures.

Table 1 reports silicon area for each of the main blocks of a router for three different routers implementing XY routing, Odd-Even routing, and table-based routing respectively. As regards the latter, two different implementations which are able to manage compressed routing tables with a budget of four (RT4) and eight (RT8) table entries have been considered (see [20] for implementation details). The cost overhead of a routing table implementation based on the proposed compression technique and architecture represent only a small fraction of the overall router cost. The overhead over a XY router is 10.9% and 23.8% for RT4 and RT8 respectively. The overhead over an Odd-Even router is 10.4% and 23.3% for RT4 and RT8 respectively.

4. Performance Evaluation

In this section we evaluate the improvement in both average adaptivity and average delay when communications concurrency information is exploited.

We generate random communication scenarios as follows. We consider randomly generated communication graphs with a given communication density ρ. (The communication density is defined as the ratio between the number of communications and the number of nodes). We model a communication scenario as a random subset of all possible communications using parameter χ for controlling

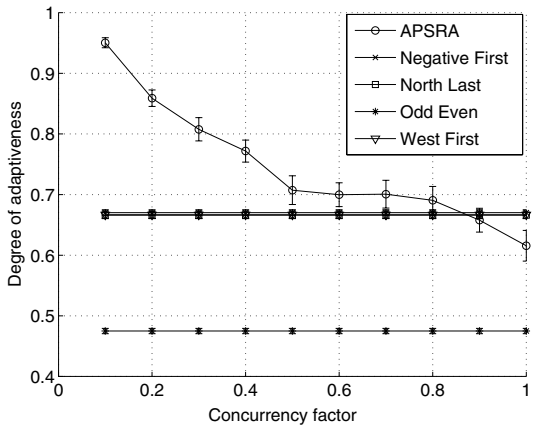


Figure 3. Adaptivity vs. communication concurrency probability for a 8×8 mesh and randomly generated communication graphs with $\rho = 2$.

the communication concurrency. χ is called *concurrency probability*. The average number of concurrent communications in an interval is given by $\chi \times (|C| - 1) + 1$. $\chi = 1$ means that all communications are concurrent whereas $\chi = 0$ means that no communications overlap in time.

Figure 3 shows the adaptivity vs. communication concurrency probability for a 8×8 NoC and randomly generated communication graphs with a communication density $\rho = 2$. We define the adaptivity like in [4]. It is essentially the number of shortest paths the routing algorithm allows from the source to the destination normalized to the total number of shortest paths. We see that APSRA adaptiveness improve very fast starting from a concurrency probability of 50%. For $\chi = 0.4$, for example, the adaptivity provided by APSRA is over 10% better than turn models and over 30% better than Odd-Even. For higher values of χ, the exploitation of communication concurrency information does not result in any significant gain in adaptiveness over turn model based routing algorithms.

To evaluate average delay we used a flit-accurate simulator developed in SystemC. We compare APSRA with both a deterministic routing algorithm (XY) and an adaptive routing algorithm (Odd-Even). We choose Odd-Even because it has been proved to exhibit the best performance among different traffic scenarios [4]. It is also the most cited adaptive routing algorithm proposed for mesh networks without using virtual channels. Recent algorithms such as DyAD [12] and contention-look-ahead routing algorithm [23] 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

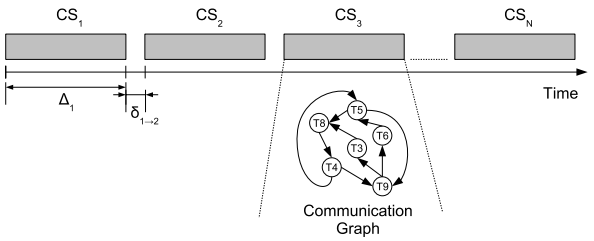


Figure 4. Managing communication concurrency through communication scenarios.

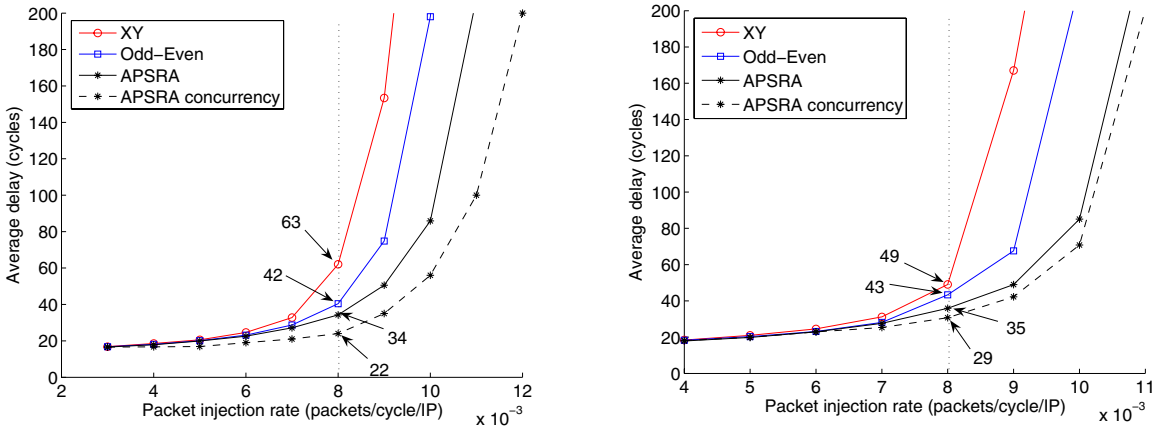
in our work. Use of virtual channels will also improve network performance for algorithms produced by APSRA. The evaluations were made on a 8×8 network using wormhole switching with a packet size randomly distributed between 2 and 16 flits. In our model, each router has an input-buffer size of 2 flits. 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.

Figure 4 gives a qualitative view about the way in which communication concurrency has been modeled in the experiments. Concurrent communications have been grouped into the same communication graph. For each communication graph, CG_i, is associated a *communication scenario* CS_i. Communication scenarios are executed in-sequence, i.e., CS_i is executed after CS_{i-1} and before CS_{i+1}. The life time of a CS_i is $|\Delta_i|$. During the interval Δ_i only the communications of CG_i are active. Between the end of the CS_i and the start of CS_{i+1} there is an inactivity period of duration $\delta_{i\rightarrow(i+1)}$. In our experiments, we set $|\Delta_i| = 50,000$ clock cycles, for $i = 1,2,3,4$ and $\delta_{i\rightarrow(i+1)} = 500$ clock cycles. The inactivity period is large enough to host both a cool down phase and a reconfiguration phase. As regards the first one, for a 8×8 mesh and 2 flits buffers, we found that a budget of 200 cycles is sufficient to flush all the flits in the network under all the traffic scenarios we considered. The reconfiguration phase requires the replacement of routing tables and can be implemented locally in each router. This is a very hot topic recently addressed by Duato *et al.* in [7] and by Lysne *et al.* in [15]. They have developed a new theory for determining deadlock properties of dynamic network reconfiguration techniques which also serves as a basis for the development of design methodologies to derive deadlock-free reconfiguration techniques [15]. Although this theory focuses on interconnection networks typically used in multiprocessor servers, network-based computing clusters, and distributed storage systems, it has a potential application also to NoCs.

Figure 5 shows average delay variation for different injection loads and for *uniform* (a) and *hot spot* traffic (b). In the uniform traffic for a source node, all other nodes have equal probability to be selected as a destination. In the hot spot traffic, the four hot spot nodes are located at the top-right corner of the NoC. We considered four non-concurrent communication scenarios randomly generated with $\chi = 0.3$. Although APSRA outperforms the other routing algorithms, an additional improvement of 29% in delay (from 34 cycles to 22 cycles) is obtained by exploiting the information about communications concurrency in the uniform traffic scenario. Compared to XY and Odd-Even, APSRA concurrency improves the average delay of 65% and 53% respecively. Similar results are obtained for the hot-spot traffic scenario with improvements in average delay ranging from 41%, 33%, and 17% over XY, Odd-Even, and APSRA respectively.

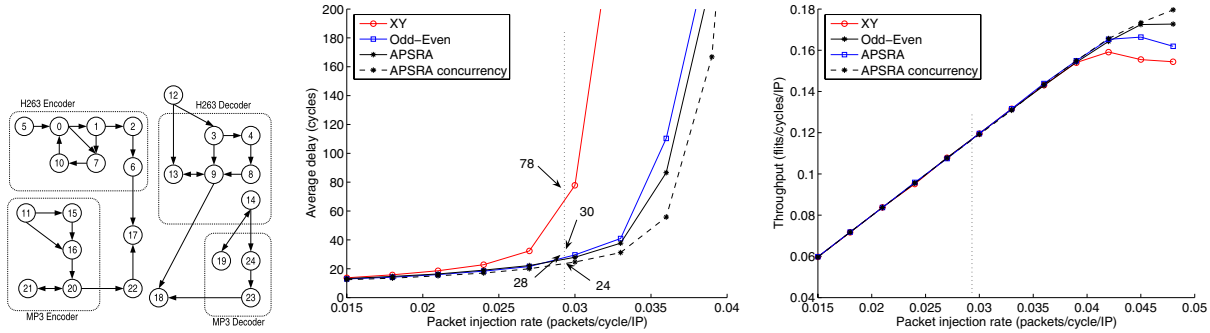
As a more realistic communication scenario we consider a generic MultiMedia System which includes an h263 video encoder, an h263 video decoder, an mp3 audio encoder and an mp3 audio decoder [11] whose communication graph is depicted in Figure 6(a). For this scenario we used self-similar packet injection distribution which has been observed in the bursty traffic between on-chip modules in typical MPEG-2 video applications [22]. Average delay and throughput variation for different injection load are reported in Figure 6(a) and Figure 6(b) respectively. For this example we consider two communication scenarios. To the first one, belongs all the communications which refer to the encoding part of the codec. To the second one, belong those which refer to the decoding part of the codec. The 25 cores implementing the system have been randomly mapped on the NoC. Adaptivity is 0.62 and 0.79 for Odd-Even and APSRA respectively. It reaches 0.98 for APSRA when communication concurrency information is considered. As can be observed, a great improvement in delay can be obtained by using adaptive routing algorithms. For an injection load of 0.03 (below saturation for all the algorithms as can be seen from the throughput curve) APSRA performs better than Odd-Even by almost 7% (28 cycles vs. 30 cycles). Using the concurrency version of APSRA improvement grows to 20% (24 cycles).

Finally, Table 2 reports a summary of the results regarding improvement in average delay obtained by APSRA with respect to other algorithms for different traffic scenarios. The average delay has been measured at the highest possible packet injection rate such that none of the algorithms show a sign of saturation. In *Locality* traffic scenario, the probability of selecting a destination node B for a source node A is inversely proportional to the distance between A and B. In *Transpose 1* and *Transpose 2* scenarios, a node (i, j) only sends packets to a node (j,i) and $(N-1-j, N-1-i)$ re-



(a) (b)

Figure 5. Delay variation under *uniform* (a), and *hot spot* traffic (b).



(a) (b) (c)

Figure 6. Communication graph of the multimedia system (a). Delay variation (b) and throughput variation (c) for a multimedia application.

Table 2. Improvement in average delay of APSRA as compared to XY and Odd-Even for different traffic scenarios.

Traffic scenario	*pir* (packets/cycle/IP)	Avg. delay (cycles)				APSRA improvement		
		XY	OE	APSRA	APSRA conc.	vs. XY	vs. OE	vs. APSRA
Uniform	0.008	63	42	34	22	65.08%	47.62%	35.65%
Locality	0.020	39	34	29	22	43.59%	35.29%	24.14%
Transpose 1	0.011	91	39	19	17	81.32%	56.41%	10.53%
Transpose 2	0.011	82	31	19	17	79.27%	45.16%	10.53%
Hotspot-4c	0.003	50	46	34	29	42.00%	36.96%	14.71%
Hotspot-4tr	0.008	49	43	35	29	40.82%	32.56%	17.14%
Hotspot-8rs	0.003	34	25	20	16	52.94%	36.00%	20.00%
Mms	0.030	78	30	28	24	69.23%	20.00%	14.29%
					Average improvement	59.38%	39.75%	18.34%

spectively, where N is the size of the mesh. *Hotspot-4c, -4tr, -8rs* refer to hot-spot traffic scenarios where hot-spot nodes are located at center, top-right and right-side of the NoC respectively. Finally, *Mms* is the traffic scenario generated by the multimedia system example. For the first seven traffic scenarios, we considered a χ value of 0.4. We observe that on average, the concurrency version of APSRA exhibits an improvement in the average delay of about 59%, 40% and 18% compared to XY, Odd-Even, and APSRA respectively.

5. Conclusions

In this paper we have made a case for using NoC paradigm for building future FPGAs in which large computational blocks will be connected to each other through packet switched networks. We proposed a methodology to develop application specific efficient deadlock free routing algorithms to specialize such general purpose devices. Our methodology not only uses the information about topology of communicating cores but also exploits information about concurrency of communication transactions. Results from analysis and simulation based evaluation demonstrate that routing algorithms developed using our approach significantly outperform general purpose routing algorithms like XY and Odd-Even for mesh topology NoC. APSRA methodology is general and can be applied to irregular networks also. In fact, in [10] we show that routing algorithm generated by the earlier version of APSRA has even higher performance and adaptivity advantage for non-homogeneous mesh NoCs.

Our methodology assumes that communication concurrency information is available after task mapping and scheduling step of system level design. Developing an interval based task mapping and scheduling methodology is an interesting research problem. Our approach also assumes configurable SRAM based tables in every router of the network. It will be interesting to investigate the performance and routability when only a very small sized router table is available in routers.

References

[1] G. Ascia, V. Catania, and M. Palesi. Multi-objective mapping for mesh-based NoC architectures. In *2nd IEEE/ACM/IFIP Int'l Conf. on Hardware/Software Codesign and System Synthesis*, pp. 182–187, Stockholm, Sweden, Sept. 2004.

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

[3] J.-M. Chang and M. Pedram. Codex-dp: Co-design of communicating systems using dynamic programming. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 19(7):732–744, July 2000.

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

[5] W. J. Dally and C. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Trans. on Computers*, C(36):547–553, 1987.

[6] J. Duato. A necessary and sufficient condition for deadlock-free routing in wormhole networks. *IEEE Trans. on Parallel and Distributed Systems*, 6(10):1055–1067, Oct. 1995.

[7] J. Duato, O. Lysne, R. Pang, and T. M. Pinkston. Part I: A theory for deadlock-free dynamic network reconfiguration. *IEEE Trans. on Parallel and Distributed Systems*, 16(5):412–427, May 2005.

[8] 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.

[9] R. Holsmark and S. Kumar. Design issues and performance evaluation of mesh NoC with regions. In *IEEE Norchip*, pp. 40–43, Nov. 2005.

[10] R. Holsmark, M. Palesi, and S. Kumar. Deadlock free routing algorithms for mesh topology NoC systems with regions. In *EUROMICRO Conf. on Digital System Design, Architectures, Methods and Tools*, pp. 696–703, Sept. 2006.

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

[12] J. Hu and R. Marculescu. DyAD - smart routing for networks-on-chip. In *ACM/IEEE Design Automation Conference*, pp. 260–263, June 2004.

[13] 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.

[14] T. Lei and S. Kumar. A two-step genetic algorithm for mapping task graphs to a network on chip architecture. In *Euromicro Symposium on Digital Systems Design*, Sept. 2003.

[15] O. Lysne, T. M. Pinkston, and J. Duato. Part II: A methodology for developing deadlock-free dynamic network reconfiguration processes. *IEEE Trans. on Parallel and Distributed Systems*, 16(5):428–443, May 2005.

[16] S. Murali, D. Atienza, L. Benini, and G. D. Micheli. A multi-path routing strategy with guaranteed in-order packet delivery and fault-tolerance for networks on chip. In *Design Automation Conference*, pp. 845–848, July 2006.

[17] S. Murali, M. Coenen, A. Radulescu, K. Goossens, and G. D. Micheli. Mapping and configuration methods for multi-use-case networks on chips. In *Asia and South Pacific Design Automation Conference*, pp. 146–151, 2006.

[18] S. Murali, M. Coenen, A. Radulescu, K. Goossens, and G. D. Micheli. A methodology for mapping multiple use-cases onto networks on chips. In *Conf. on Design, Automation and Test in Europe*, pp. 118–123, 2006.

[19] M. Palesi, R. Holsmark, S. Kumar, and V. Catania. A methodology for design of application specific deadlock-free routing algorithms for NoC systems. In *Int'l Conf. on Hardware-Software Codesign and System Synthesis*, pp. 142–147, Oct. 2006.

[20] 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*, pp. 373–384, July 2006.

[21] E. Rijpkema, K. G. W. Goossens, A. Radulescu, J. Dielissen, J. van Meerbergen, P. Wielage, and E. Waterlander. Trade offs in the design of a router with both guaranteed and best-effort services for networks on chip. *IEE Proc. of Computer Digital Techniques*, 150(5):294–302, Sept. 2003.

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

[23] 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.