Bandwidth Aware Routing Algorithms for Networks-on-Chip

G. Longo[a], S. Signorino[a], M. Palesi[a,*], R. Holsmark[b], S. Kumar[b], and V. Catania[a]

[a] Department of Computer Science and Telecommunications Engineering
University of Catania, Italy

[b] Department of Electronics and Computer Engineering
School of Engineering, Jönköping University, Sweden

[*] *Corresponding author*: Maurizio Palesi (mpalesi@diit.unict.it)

***Abstract*— In this paper we present a technique to design topology-agnostic highly adaptive bandwidth-aware application specific deadlock-free routing algorithms for networks-on-chip (NoC). The basic idea relies on the exploitation of the designer knowledge of communication traffic characteristics (like communication topology and communication bandwidth), to customize the routing algorithm. The routing algorithms designed using our approach ensure that limits on link bandwidth are not violated. Through experiments we show that the routing algorithms also distribute traffic more uniformly on links.**

I. Introduction

One of the main factors which affects the overall performance of a Network-on-Chip (NoC) is represented by the routing algorithm [1]. Routing algorithm determines the path selected by a packet to reach its destination.

Traditionally, routing algorithms have been designed without any reference to the characteristics of the traffic which would stimulate the network. The main reason was that, in a general purpose domain, the communication traffic cannot be accurately characterized, thus the routing algorithm has to be designed to work properly for any traffic. As a consequence, the design of the routing algorithm is generally carried out in a very conservative way, assuming that all the network node pairs need to communicate. However, in the application specific domain, which characterizes the area of embedded systems, an accurate characterization of the communication traffic is often possible. The embedded system designer has an in depth knowledge of the application (or the set of applications) which will be mapped on the system. This knowledge opens some new directions in system optimization like, for instance, the customization of the routing algorithm for a given application.

Based on this, in [2] we presented APSRA, a methodology to design application specific routing algorithms for NoC systems. Starting from the knowledge of the communication topology (i.e., which are the network nodes that communicate and which do not), the goal of APSRA is to maximize the degree of adaptiveness of the routing function and guarantee deadlock freedom.

Unfortunately, APSRA does not take into account the communication attributes like the communication bandwidth requirements of the different task pairs mapped on different network nodes. Thus, the selection of the routing paths to be removed to restrict the routing function and guarantee deadlock freedom, is carried out in a blind fashion. That is, it is implicitly assumed that all the communications have the same bandwidth requirements. Such unawareness may lead to a bad distribution of the traffic load over the network. This is particularly true when the range of the bandwidth requirements of different communications is large. Unfortunately, this is a very frequent case in real applications. In [3], for example, the range of communication bandwidth requirements for a Video Object Plane decoder in a MPEG-4 decoder system spans from 16 MB/s to 500 MB/s.

The above considerations motivated this work. As the traffic characteristics of a routing path are generally different from those of another routing path, they should be treated differently. For this reason, we believe that emphasizing the role of communication bandwidth requirements during the design of the routing algorithm adds a new degree of freedom in system performance optimization.

II. Problem Formulation

Simply stated, for a given application and a given network topology, the goal is to generate a routing algorithm which is strongly adaptive and spreads the traffic over the network in such a way that the communication traffic of any link will not exceed its capacity (maximum sustainable bandwidth). To formulate the problem more formally, we define the following terms.

The *Communication graph*, $CG = G(T,C)$, is a directed graph, where T is the set of tasks and C is the set of communications. For each communication $c_{i,j} = (t_i, t_j) \in C$, task $t_i \in T$ denotes source task, and task $t_j \in T$ denotes destination task. For a communication $c \in C$, the function $B(c)$ returns the bandwidth requirement, that is the minimum bandwidth that should be allocated by the network in order to meet the performance constraints for communication c.

This work was supported by the European Commission in the context of the HiPEAC network of excellence (project 004408 and 217068), under the Interconnects research cluster on High Performance Interconnection Networks for Embedded Applications.

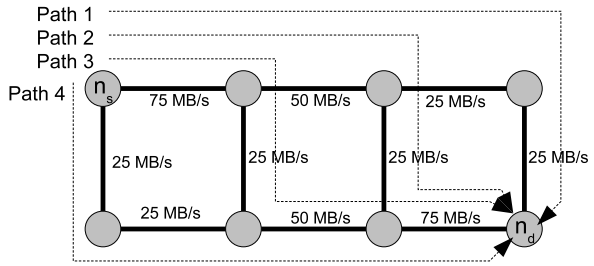

Fig. 1. Effective bandwidth for a communication from node n_s to node n_d at 100 MB/s assuming a fully adaptive minimal routing.

The *Topology graph*, $TG = G(N,L)$, is a directed graph which models the network topology. N is the set of network's nodes, and L is the set of network's channels. Channel $l_{i,j} = (n_i, n_j)$ connects node $n_i \in N$ to node $n_j \in N$. Given a channel $l \in L$, the function $Cap(l)$ returns its capacity.

The *Mapping function*, $M : T \rightarrow N$, maps tasks to network nodes (e.g., if $M(t_i) = n_j$ then task t_i is mapped on node n_j of the network).

As we are dealing with adaptive routing, the required bandwidth for communication c is split over multiple paths that the routing function allows for that communication. For the sake of example, consider Figure 1 which shows a 4×2 mesh-based network topology. Let us suppose that communication $c = (t_s, t_d)$ requires a bandwidth of 100 MB/sec (load) and that the routing function allows all the minimal paths from node $n_s = M(t_s)$ to node $n_d = M(t_d)$ (four paths in total). The load is distributed over the paths as shown in Figure 1 which reports, for each network channel, the *effective bandwidth* (or *effective load*) (EB) and the total number of paths containing that channel. Formally, the effective bandwidth of a channel $l \in L$ due to a communication $c \in C$ can be computed as:

$$EB(c,l) = B(c) \times \frac{|PT(c,l)|}{|\mathscr{P}(c)|}$$

where $\mathscr{P}(c)$ denotes the set of minimal paths admitted by the routing function for communication c, and $PT(c,l) = \{P \in \mathscr{P}(c) : l \in P\}$ is the *pass through link* set, that is the set of paths of c which contain the link l. Finally, we indicate with $AB(l)$ the *aggregate bandwidth* of l which is computed as:

$$AB(l) = \sum_{c \in C} EB(c,l).$$

Using these definitions, the bandwidth-aware routing algorithm problem can be formulated as follows. Given a communication graph CG, a topology graph TG and a mapping function M, find a routing function R which is deadlock-free such that:

$$\forall l \in L \Rightarrow AB(l) \leq Cap(l), \quad (1)$$

that is, the communication load of any channel, l, must not exceed its capacity $Cap(l)$.

III. Bandwidth Aware Routing Algorithm

Given a routing function R for a network topology $TG(N,L)$, it is possible to build the *channel dependency graph* (*CDG*) [4]. The *CDG* is a directed graph whose nodes are the network's channels and edges represents direct dependencies between these channels. There is a direct dependency between two channels l_i and l_j if l_j can be used immediately after l_i by messages destined to some node $n \in N$. For instance, for the 2×2 mesh-based network shown in Figure 2(a), let us suppose that *xy* routing is used. In this case there is a direct dependency from channel $l_{0,1}$ to channel $l_{1,3}$, as *xy* allows that for communication from node n_0 to node n_3, channel $l_{1,3}$ can be used immediately after channel $l_{0,1}$. On the contrary, there is not a direct dependency from, for example, $l_{3,1}$ to $l_{1,0}$ as *xy* prohibits that messages are routed along *y*-direction before they are routed along *x*-direction. Figure 2(b) shows the *CDG*. In [5] Duato proved that if the *CDG* is acyclic then R is deadlock free.

In [2] we introduced the concept of *application specific channel dependency graph* (*ASCDG*). The *ASCDG* is a subgraph of the *CDG* obtained by exploiting communication information. Let us consider again Figure 2(a). If we are aware that task mapped on node n_0 will never communicate with task mapped on node n_3, there will not be messages using the path $l_{0,1} \rightarrow l_{1,3}$, thus there is no application specific channel dependency from $l_{0,1}$ to $l_{1,3}$. Figure 2(c) shows the *ASCDG* built considering that task mapped on n_0 never communicate with task mapped on n_3 and task mapped on n_2 never communicate with task mapped on n_1. As the number of application specific channel dependencies is less than or equal to the number of channel dependencies, the number of cycles (if any) the *ASCDG* will be less than that in the *CDG*.

If the *ASCDG* contains cycles, in [2] we proposed an heuristic to break all the cycles in such a way that the loss in adaptivity is minimized. In the next subsections, we present a new technique to break cycles in the *ASCDG* which is bandwidth-aware. That is, communications bandwidth information is used to select the application specific channel dependency to be removed with the aim to uniformly distribute the traffic over the network. More precisely, the basic idea is to allocate more routing paths (i.e., give more adaptivity) to communications characterized by higher communication bandwidth. In addition, we propose a recovery procedure that reallocates communication paths in such a way that communication load do not exceed network's channels capacity.

A. Breaking Cycles of the ASCDG

A cycle in the *ASCDG* is a succession of application specific direct dependencies $D = \{d_1, d_2, \ldots, d_n\}$, where a $d \in D$ is a pair (l_i, l_j) with $l_i, l_j \in L$. Here the problem is the selection of the best dependency to be removed to break the cycle D. Remove a dependency means to remove all the paths which determine that dependency. As soon as a path is removed, the fraction of bandwidth it transports must be redistributed between the remaining paths. The idea we propose in this paper is based on removing the dependency d which minimizes the overhead of bandwidth that should be allocated to the remaining paths that do not determine the dependency d.

Formally, let us indicate with $PT^2(c,d)$ the *pass through dependency* set, that is the set of paths of c which determine

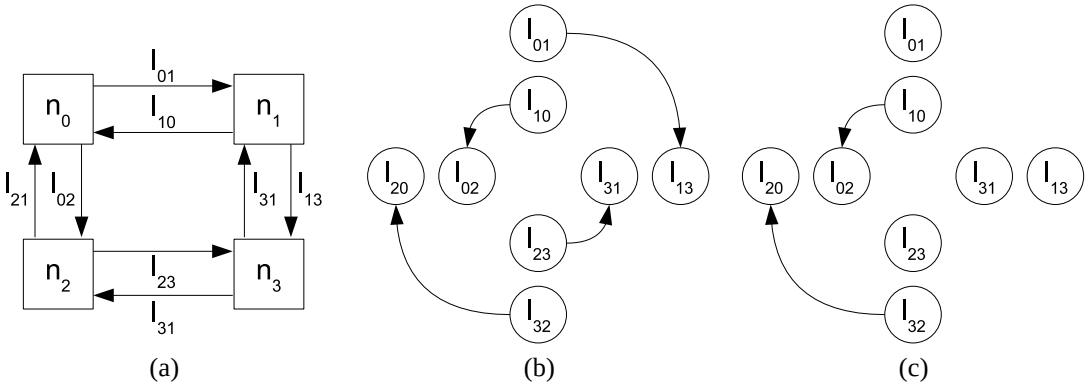

Fig. 2. A 2×2 mesh based network topology (a). The channel dependency graph (b) for the *xy* routing function and the application specific channel dependency graph (c) if node n_0 never communicates with node n_3 and node n_2 never communicates with node n_1.

```
1  BreakCycles(ASCDG: application specific CDG,
2              C: set of communications,
3              P: set of admissible paths) {
4    while ASCDG is not acyclic do {
5      D ← GetCycle(ASCDG);
6      cost ← ∞;
7      for d ∈ D {
8        if ¬∃ c ∈ C : P(c) \ PT²(c,d) = ∅ {
9          cost' ← Σ_{c∈C} B(c)×|PT²(c,d)| / (|P(c)|×(|P(c)|−|PT²(c,d)|));
10         if cost' < cost {
11           cost ← cost';
12           d' ← d;
13         }
14       }
15     }
16     ASCDG ← ASCDG \ {d'};
17     P(c) ← P(c) \ ∪_{c∈C} PT²(c,d');
18   }
19 }
```

Fig. 3. Break cycles algorithm.

the dependency $d = (l_1, l_2)$:

$$PT^2(c,d) = PT(c,l_1) \cap PT(c,l_2).$$

Let d be an application specific direct dependency. To remove d all the paths of any communication c which determine d must be removed. For communication c the aggregated bandwidth to be redistributed is $[B(c)/|\mathscr{P}(c)|] \times |PT^2(c,d)|$. This bandwidth is redistributed between the $|\mathscr{P}(c)| - |PT^2(c,d)|$ remaining paths which do not determine the dependency d. Based on this, the dependency to be removed is the $d \in D$ such that the cost function:

$$cost(d) = \sum_{c \in C} \frac{B(c) \times |PT^2(c,d)|}{|\mathscr{P}(c)| \times (|\mathscr{P}(c)| - |PT^2(c,d)|)} \qquad (2)$$

is minimized. This ensures that the dependency which will be chosen for removal is such that the load on the paths which determine that dependency is redistributed on the maximum possible number of alternative paths.

The breaking cycles algorithm is reported in Figure 3. For each cycle of the *ASCDG*, only the channel dependencies that, if removed, do not cause reachability problems, are considered. This check is performed by ensuring that there does not exist any communication whose all routing paths determine such channel dependency (line 8). Thus, the channel dependency, d', which minimizes the cost function (2) is selected and removed from the *ASCDG* (line 16). Then, all the routing paths which determine d' are removed from the set of admissible paths (line 17).

B. Bandwidth Reallocation

Using the procedure discussed in the previous subsection, we obtain a routing function which is deadlock free (as the *ASCDG* is acyclic) and which generates a set of routing paths by providing more adaptivity to communications characterized by higher communication bandwidth. However, it is possible that the aggregate bandwidth on some network links exceeds the capacity of that links [i.e., Condition (1) is not satisfied for some $l \in L$]. In this case, some routing paths passing through those links, must be removed to reduce the aggregate bandwidth on those links down to the links capacity or, more in general, down to a user defined value.

The proposed algorithm is shown in Figure 4. The input parameters are the set on network links, the set of communications, the set of admissible paths derived from the procedure described in the previous subsection, and a threshold which defines the maximum bandwidth which has not to be exceeded in any network link. The output is the updated set of routing paths. The procedure starts by sorting network links in descending order based on their aggregate bandwidth. For each link l and for each communication c which has at least one path using l, but has more than one path, two lists named *paths2rem* and *paths2enr* are generated as follows. *path2rem* contains all the paths for c that should be removed as they use network links whose load exceed the threshold. *paths2enr* contains those paths that can be used by other communications (i.e., can be enriched) as they use links whose load is below the threshold. Then, the list *paths2rem* is scanned and routing paths belonging to it are removed from $\mathscr{P}$. Of course, removing a path causes the redistribution of the bandwidth allocated on it on the other paths belonging to *paths2enr*. Thus, the path elimination stops when there is at least one path in *paths2enr* that contains a link whose load exceeds the threshold. The above steps are repeated until the load on each link does not exceed the threshold. This

```
1  BandwidthRealloc(L: set of links,
2                   C: set of communications,
3                   P: set of admissible paths,
4                   threshold: integer) {
5   while ∃ l ∈ L : AB(l) > threshold do {
6    rem_flag ← false;
7    Sort links in L = {l_1, l_2, ..., l_n} in descending
8     order of load. That is, such that
9     AB(l_i) > AB(l_{i+1}), i = 1,2,...,n−1;
10   for l ∈ L {
11    for c ∈ C : PT(c,l) ≠ ∅ {
12     paths2rem ← paths2enr ← ∅;
13     if |P(c)| > 1 {
14      for P ∈ P(c) {
15       if ∃ l ∈ P : AB(l) > threshold
16        paths2rem ← paths2rem ∪ {P};
17       else
18        paths2enr ← paths2enr ∪ {P};
19      }
20      if paths2enr ≠ ∅ {
21       rem_flag ← true;
22       for P ∈ paths2rem {
23        P(c) ← P(c) \ {P};
24        if ∃ P' ∈ paths2enr : ∃ l' ∈ P' : AB(l') > threshold
25         break;
26       }
27      }
28     }
29    }
30   }
31   if ¬rem_flag {
32    show("Not able to reallocate paths to
33          meet the required threshold.");
34    return;
35   }
36  }
37 }
```

Fig. 4. Bandwidth reallocation algorithm.

procedure aborts if the path elimination step cannot be carried out due to reachability issues which arises when it needs to remove a path which is unique for a certain communication.

IV. Experiments and Results

We evaluate the proposed approach on both synthetic and real traffic scenarios. As synthetic traffic scenarios, we consider [6] *uniform*, *transpose*, *bit-reversal*, *shuffle*, *butterfly*, and *hot-spot*. For them the bandwidth for each communicating pair has been randomly generated between 10 and 100 MB/sec. As a more realistic communication scenario we consider a generic MultiMedia System which includes an H.263 video encoder, an H.263 video decoder, an mp3 audio encoder and an mp3 audio decoder [7]. The application is partitioned into 40 distinct tasks which have been manually mapped on a 5×5 mesh-based NoC.

In the following we indicate with APSRA the approach proposed in [2], with APSRA-BW the variant of APSRA using the heuristic presented in Subsection III-A, and with APSRA-BWL the augmented version of APSRA-BW with the bandwidth reallocation procedure discussed in Subsection III-B.

TABLE I

Percent decrease of standard deviation of the aggregated bandwidth in network links.

Traffic	APSRA-BW	APSRA-BWL
Uniform	25%	27%
Bit-reversal	19%	23%
Butterfly	0%	2%
Shuffle	18%	19%
Transpose1	0%	2%
Transpose2	0%	2%
Hot-spot_C	10%	12%
Hot-spot_TR	5%	10%
MMS	5%	5%
Average	10%	12%

Let us start by showing the effectiveness of the proposed approach in uniformly distributing the traffic over the network. To do this, we use the standard deviation of the aggregate bandwidth in the network links as the evaluation metric. Using this metric, we compare APSRA, APSRA-BW, and APSRA-BWL on a 8×8 mesh based NoC under different traffic scenarios. For the APSRA-BWL, we fix the threshold to the 90 percent of the maximum aggregate bandwidth when fully adaptive minimal routing is used. For each traffic, Table I reports the percentage decrease of standard deviation of the aggregated bandwidth in network links when both APSRA-BW and APSRA-BWL are used. As can be seen, the proposed heuristic to break cycles of the *ASCDG* allows to better distribute the bandwidth across the network. There are some situations, in which there is not any reduction in standard deviation. This is the case of *transpose* and *butterfly* traffics in which the *ASCDG* is already acyclic and the cutting edge heuristic does not take place. On average the standard deviation of the aggregated bandwidth in network links decreases by 10%. An additional improvement of 2% is obtained when the bandwidth redistribution procedure is used. On the other side, as discussed in Subsection III-B, the elimination of some routing paths operated by the bandwidth redistribution procedure, negatively affects the adaptiveness of the routing function as shown in Figure 5. As can be see, for *butterfly* and *transpose* traffic, there is no loss in adaptivity due to the fact that for such traffics the *ASCDG* is acyclic, thus the heuristic to break cycles does not take place. It is interesting to observe that, for some traffics, like *bit-reveral* and *shuffle*, the adaptivity of APSRA-BW is higher than that of APSRA. Although the main objective of APSRA is the maximization of adaptivity, the heuristic to break cycles immediately stops when the first solution is found.

Figure 6 shows the aggregate bandwidth of any link of a 9×9 mesh based NoC under *uniform* traffic for both the routing algorithm generated by APSRA and by APSRA-BWL. The threshold has been fixed to 550 MB/sec. As can be observed, when APSRA is used, the aggregate bandwidth in several link exceeds the threshold. If this threshold represents the network link capacity, such bandwidth overheads translate in local network congestion that, due to back pressure mechanism along with the wormhole switching technique, propagates to the entire network causing a strong degradation of overall

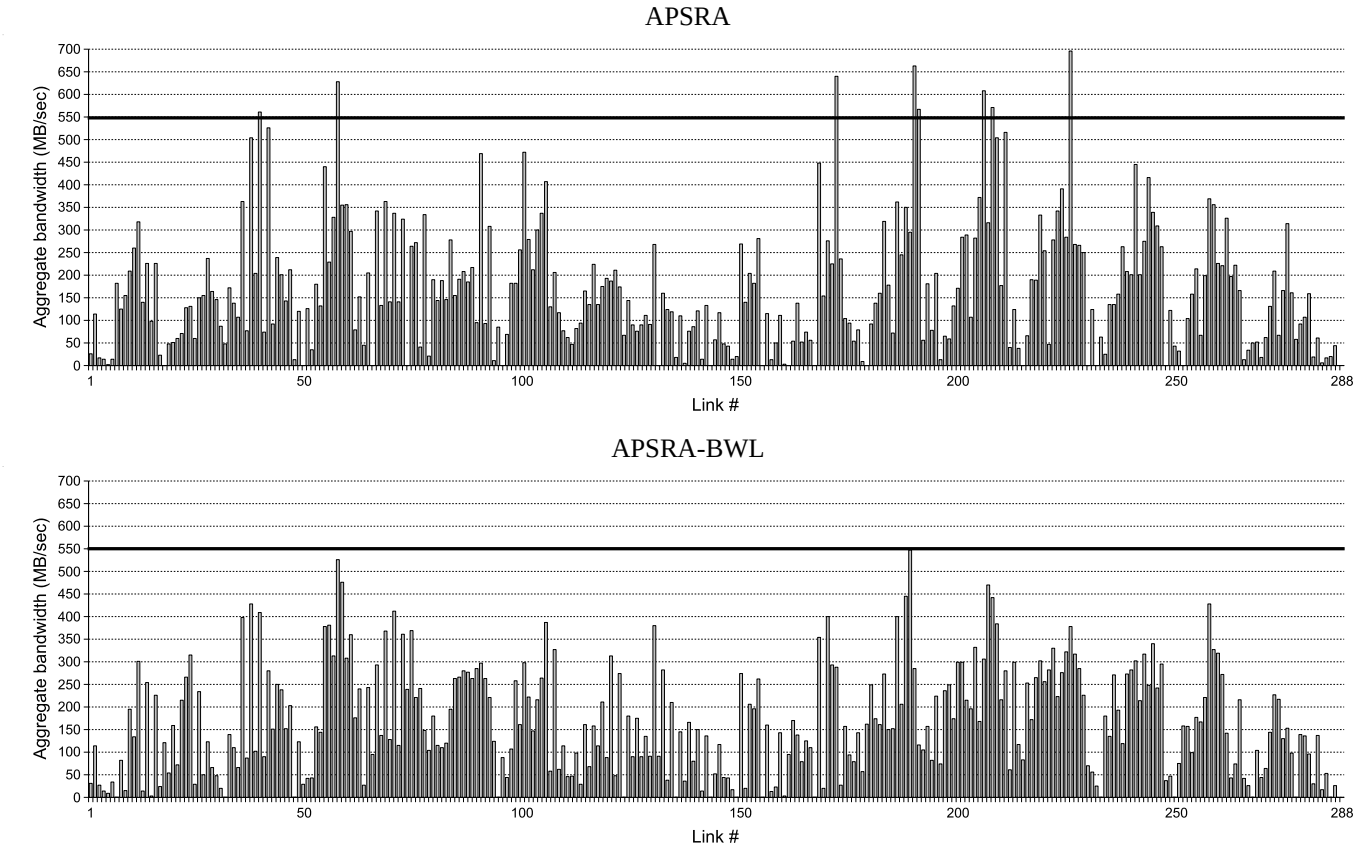

Fig. 6. Aggregate bandwidth per link for a 9×9 mesh-based NoC under *uniform* traffic. Routing algorithm used is generated by APSRA (top) and APSRA-BWL (bottom).

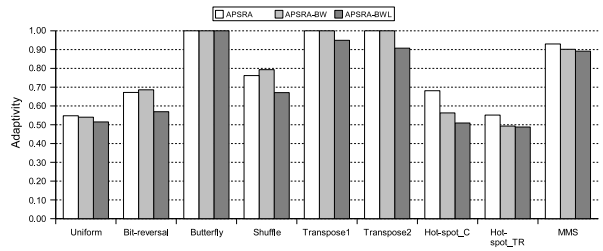

Fig. 5. Adaptivity of the routing algorithms generated by APSRA, APSRA-BW, and APSRA-BWL.

network performance.

V. Conclusions

In this paper we have presented a new approach to design topology-agnostic high-adaptive bandwidth-aware application specific deadlock-free routing algorithms for NoCs. The basic idea behind the approach is the exploitation of communication bandwidth information to customize the routing algorithm for a given application. The approach is divided in two phases. In the first phase all the cycles of the ASCDG are removed by means of an heuristic whose main objective is to uniformly spread the load over the network. Then, in the second phase, the routing function is restricted by removing all the routing paths which determine the overload of some network links. The approach has been evaluated on both synthetic and real traffic scenarios. The results obtained show that the routing function generated by the proposed approach i) is highly adaptive, ii) reduces the variation of load in the network links, iii) ensures that the link bandwidth limit is not violated. Since the proposed ideas are improvements over APSRA, a table based router is the best implementation option for the routing function [2].

References

[1] U. Y. Ogras, J. Hu, and R. Marculescu, "Key research problems in noc design: A holistic perspective," in *International Conference on Hardware-Software Codesign and System Synthesis*, Sept. 2005, pp. 69–74.

[2] M. Palesi, R. Holsmark, S. Kumar, and V. Catania, "A methodology for design of application specific deadlock-free routing algorithms for NoC systems," in *International Conference on Hardware-Software Codesign and System Synthesis*, Seoul, Korea, Oct. 22–25 2006, pp. 142–147.

[3] E. B. van der Tol and E. G. Jaspers, "Mapping of MPEG-4 decoding on a flexible architecture platform," *Media Processors*, vol. 4674, pp. 362–375, 2002.

[4] W. J. Dally and C. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Transactions on Computers*, vol. C, no. 36, pp. 547–553, 1987.

[5] J. Duato, "A necessary and sufficient condition for deadlock-free routing in wormhole networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 6, no. 10, pp. 1055–1067, Oct. 1995.

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

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