

# Using Multiple Paths for Guaranteed Resource Allocation and Improved Best Effort Performance in NoCs

I. Ovadia, Y. Ha, H. Corporaal

**Abstract**— Networks-on-Chips (NoCs) provide communication platforms to Systems-on-Chips (SoCs). In NoCs, channels are generally shared between traffic flows, resulting in contention. However, certain flows require delivery guarantees. Differentiated quality-of-service (QoS) is achieved by providing guaranteed services such as guaranteed throughput (GT) to certain flows, on top of the regular best-effort (BE) delivery. Most current design methodologies employ a single-path mapping for each traffic flow. Such resource allocation is suboptimal for GT traffic and severely degrades performance of BE traffic. To solve this problem, we propose spatially distributing traffic and its guarantees in several paths per flow, while keeping the BE routing simple. Our experiments on a mesh in a transaction level simulator indicate that improvements of 20-48% in latency and 6-20% in throughput of BE traffic are achievable. We argue that this gain can be achieved cheaply with simple modifications to current methodologies.

**Index Terms**—Multiprocessors interconnection, Network-on-chip, quality of service, guaranteed throughput, best effort

## I. INTRODUCTION

VLSI System-on-Chip (SoC) design complexity is continuously increasing. While current systems comprise tens of modules, this is expected to grow by at least an order of magnitude in the coming years. To manage such design complexities it is necessary to decouple the design of computation and communication, which entails platform-based or communication-centric methodologies. Such methodologies include the use of on-chip networks, resulting in a so-called Network-on-Chip (NoC). A regular NoC approach is advantageous in terms of IP reuse, interconnection and communication reuse, and thus in meeting stringent time-to-

market constraints. NoCs are a promising alternative to bus-based approaches for many reasons including wiring, power consumption, and scalability [1-8,17].

We define a *flow* as a complete data transfer between a source and a destination. In a NoC traffic flows generally share links, leading to contention. However, many data flows of typical (e.g. multimedia) applications require delivery guarantees. Traffic is then divided into classes to which the network provides class-differentiated Quality-of-Service (QoS). Research has been conducted with two traffic classes: *Guaranteed Throughput* (GT) and *Best Effort* (BE). In the case of GT, packets are delivered with quality guarantees, provided that their producer complies with certain restrictions on packet generation. These quality guarantees pertain to, for example packet latency, bandwidth, loss, etc. On the other hand, BE traffic is not restricted, but the network only makes its best delivery effort. As a result, BE traffic performance is strongly affected by other traffic in the network, while GT traffic enjoys delivery guarantees. [9, 20]

Current NoCs provide guarantees by reserving resources along a single path through the network per GT flow. This may create bottlenecks in the network by severely obstructing BE and limiting other GT traffic. This argument still holds when programming occurs per period rather than for the entire duration of the flow, as is the case in some networks.

To solve the above problem, we propose programming several minimal alternative paths per guaranteed flow during the static configuration of that flow. This is effectively a spatial distribution of traffic and its guarantees, which lowers the load per path. We show that combined with simple BE routing, this approach significantly improves the performance of BE traffic.

The rest of the paper is organized as follows. Related work is described in section 2. The problem and our solution are described in section 3. Section 4 gives the details of a simulation we have conducted, and analyzes its results. Section 5 gives a more generic evaluation of the approach and a cost analysis. Finally, section 6 concludes this paper.

## II. RELATED WORK

In this section we consider several examples of NoCs and their guaranteed services. For the spectrum of possible

Manuscript received September 9<sup>th</sup>, 2005.

I. Ovadia's internship at the National University of Singapore and the Design technology Institute in Singapore resulted in this paper. He is currently working on his Master thesis within the  $\mathcal{A}$ Ethereal project at Philips Research and Silicon Hive in Philips High Tech Campus in Eindhoven. He expects to graduate from Eindhoven University of Technology this year (email: I.Ovadia@student.tue.nl)

Y. Ha is associate professor at the Electrical Engineering department of the National University of Singapore (e-mail: elehy@nus.edu.sg).

H. Corporaal is professor at the Electrical Engineering Department, Eindhoven University of Technology and is research director of the Design Technology Institute, Eindhoven. (e-mail: H.Corporaal@tue.nl).

guaranteed services implementations see [20]. In this paper we consider solutions that allocate resources to specific flows in space and time, in a form of time-division multiplexing. The transfer granularity discussed next is in *flits*, rather than entire packets.

As a first example we consider the Technion's **QNoC** (QoS NoC) that is based on a 2-D mesh. Pre-emptive priority scheduling provides timing predictability between four service classes, and round robin scheduling is used within a class. Since e-cube wormhole routing is employed, flits of the same flow traverse the same single path [14-16].

Philips' **Ætheral** provides a GT service on top of a best-effort service, and uses source routing on a mesh. A static schedule combined with a prioritization mechanism provides timing predictability for GT traffic. Again, a single GT path (here called *connection*) is used per flow [9-13].

In the **Nostrum** mesh network hot-potato routing is used for best effort traffic. GT traffic is facilitated by zero-payload best-effort packets moving back and forth between the source and destination. They essentially reserve bandwidth for payload through a single path in each direction, which is called a *virtual circuit* [17-19].

The services in the above examples statically reserve resources in a single path per flow. The situation may be improved by using more dynamic optimizations, when the instantaneous load is lower than reserved. QNoC and Ætheral, for example, allow lower-priority (best-effort) traffic to use a resource when a reservation is temporarily not required. This is an immediate and local solution, in the sense that it pertains to each resource individually.

The service level of Nostrum, as another example, can be reduced in real time, and increased (non-real time). In [19], a method to temporarily free resources by informing them of demand slack in advance is explored on the Nostrum NoC. This method requires knowledge of demand in advance, but frees all resources along the path. Furthermore, it works in real-time because GT packets pass slack-time information.

Thus we observe that dynamic optimizations can be built on top of static GT solutions, but all cases use single path reservation. It is worth investigating the effects of increasing path diversity since it is load balancing compared to a single-path approach [20], and has been extensively used in interconnection networks like the Internet [22-23]. Regarding NoCs in particular, [24] considers GT traffic only, and shows that a multi-path approach performs better than a single-path approach because it lowers the bandwidth requirement per resource.

### III. APPROACH

Current NoC methodologies provide delivery guarantees by reservation of resources along a single resource path per GT flow. This restricts our freedom in providing guarantees to other potential GT flows. Furthermore, it is heavily obstructive to best effort traffic, thereby degrading overall BE performance.

Adaptive routing and/or flow control solutions only offer a partial solution to the problem, and are expensive to implement. The currently employed dynamic optimizations on top of static services, as outlined in the previous section, have a spatially limited impact. Thus we seek a more comprehensive approach to improve performance.

Our approach reads the following:

*Use several paths per GT flow on your NoC to distribute the load between them, while keeping the BE routing simple.*

We thus extend De Micheli's approach ([24]) by recommending simple BE routing. That is, we investigate how using a multi-path configuration for a high ranking traffic class affects a lower-ranking traffic class. We claim that the approach has positive, cost-effective effects on BE performance. Furthermore, dynamic and adaptive approaches can still be used on top of our approach. In the following sections we perform simulations to prove our approach and evaluate it with a cost analysis. We also discuss how the approach can be implemented cheaply on at least one platform.

### IV. SIMULATION

Current NoC methodologies provide delivery guarantees by reservation of resources along a single resource path per GT flow. This restricts our freedom in providing guarantees to other potential GT flows. Furthermore, it is heavily obstructive to best effort traffic, thereby degrading overall BE performance.

Adaptive routing and/or flow control solutions only offer a partial solution to the problem, and are expensive to implement. The currently employed dynamic optimizations on top of static services, as outlined in the previous section, have a spatially limited impact. Thus we seek a more comprehensive approach to improve performance.

Our approach reads the following:

*Use several paths per GT flow on your NoC to distribute the load between them, while keeping the BE routing simple.*

We thus extend De Micheli's approach ([24]) by recommending simple BE routing. That is, we investigate how using a multi-path configuration for a high-ranking traffic class affects a lower-ranking traffic class. We claim that the approach has positive, cost-effective effects on BE performance. Furthermore, dynamic and adaptive approaches can still be used on top of our approach. In the following sections we perform simulations to prove our approach and evaluate it with a cost analysis. We also discuss how the approach can be implemented cheaply on at least one platform.

#### A. Simulation Model

A 4x4 mesh topology model with E-cube routing has been developed in [21]. Two traffic classes are supported, namely BE and GT, with the latter enjoying absolute priority over the former. In the sequel, a simulator time unit is referred to as 1 second and we consider the transportation of *flits*, i.e. parts of packets.

Figure 1 (next page) shows the topology with the 16 *tiles*,

each connected to immediate neighbors via bi-directional *communication links* over which flits are transported. The links have a delay of 0.0015 seconds per flit, which translates to a bandwidth of 670 f/s. The particular configuration details are given in section 4.3.

The structure of a tile is as follows: The connections visible in Figure 1 link between *routers* inside the particular tiles. *E-cube routing* policy is employed, and uses prioritized output queuing depending on the traffic class. Thus, BE queue has size 5 and GT queue size 2. The router is modeled such that each 0.001 second a flit from the appropriate queue is released in each direction. Thus the GT queue must be empty before the BE queue is used. Therefore when a resource is not occupied by GT flits, BE flits may use the available bandwidth.

The router of each tile connects to a *Network Interface* (NI) with a bandwidth of 670 f/s, which in turn connects to a *traffic generator* through a bandwidth of 1000 f/s. The traffic generator accepts best-effort or GT traffic, but generates only one of the two. Details on traffic generation are given in section 4.2. The NI buffers flits coming from the tile (buffer size 50) and sends each flit to its destination. BE flits have a single (final) destination. In the GT case, the NI programs a particular path using several *intermediate destinations*, which the GT-flit then traverses sequentially. These figures were based on experience with NoCs.

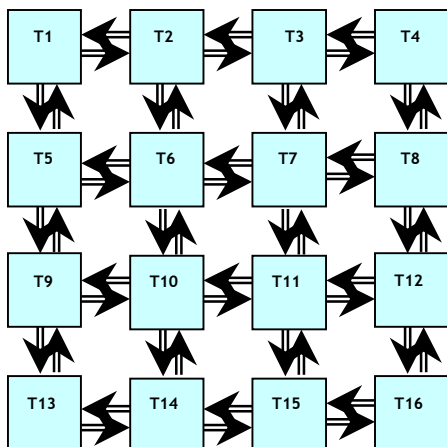


Figure 1: System schematic; there are 16 tiles in total.

### B. Details of Traffic Models

This section describes and demonstrates the BE and GT traffic models used for the traffic generators. On our mesh both BE and GT traffic are bursty. Within a burst a flit is either launched, or the burst ended, according to a Bernoulli distribution. This Bernoulli distribution is parameterized such that the mean burst length is 15. The expected burst size is given by:

$$ExpBurstSize = 1/(1-p_{next}),$$

where  $p_{next}$  represents the packet injection probability [20]. Solving for  $ExpBurstSize=15$  yields:

$$p_{next} = (MeanBurstSize-1)/MeanBurstSize = 14/15$$

That is, a next packet is launched with a probability  $14/15$ , and the burst ends with probability  $1/15$ . We would like to specify the *load* of the stream as a decimal fraction representing the average fraction of time that the source attempts to inject packets into the network. This is achieved using an exponential distribution with mean waiting time given by  $1/\lambda$  where:

$$\lambda = (load/(1-load))*LinkCapacity*(1-p_{next}),$$

with  $LinkCapacity = 1000$ .

For example, when  $load=0.4$ , an average node would attempt to inject a flit into the network 40% of the time. The load does not reflect the traffic offered; this must be measured!

The traffic pattern has been analyzed in order to verify this behavior. An example with an 80% load is presented in Figure 2. It plots an instance of 30000 time units, in which each time a packet is generated a '1' is plotted, and each time that it is not generated a '0'. Time, in 1000s of cycles, is plotted horizontally. When a moving average is plotted, notice the average is hovering around 0.8. A mathematically calculated average of this sample equals 0.792.

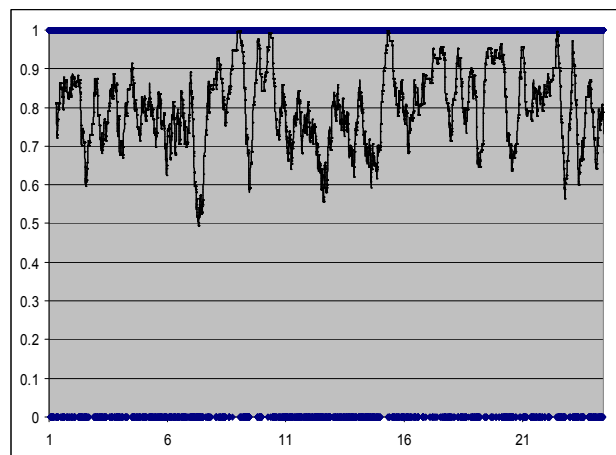


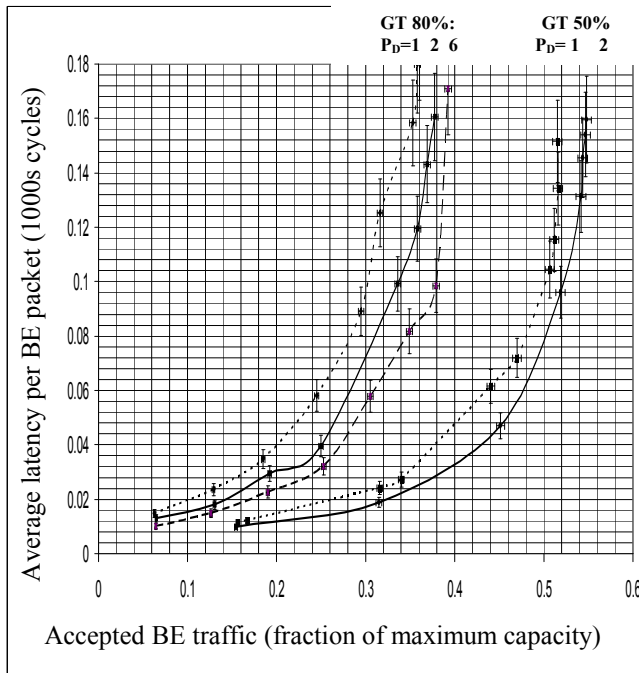
Figure 2: Illustration of traffic pattern under 80% load.

### C. Experiment

In this experiment we only consider a single GT data flow, **from nodes 6 to 16**. A path  $A$  is considered *different* from path  $B$  if it contains at least one hop that path  $A$  does not. We define the *GT Path Diversity* ( $P_D$ ) as the number of alternative flit paths over which the GT data flow is communicated to the destination. In this experiment, we further consider only the 6 different *minimal* paths between nodes 6 and 16. This is an instance of a minimal oblivious routing [20]. To simulate the network with  $P_D=1$ , the NI of node 6 is programmed to switch between the available paths every 10000 packets. For higher path diversities, the NI switches between the appropriate path *combinations* every 10000 packets. We measure network parameters for varying offered BE traffic measured as a fraction of the maximum throughput:  $\Theta_{MAX} = 4b/k = 4*(1/0.0015)/4 = 670$  f/s, with  $b$  the channel bandwidth and  $k$

the topology radix [20]. Note that each measurement point was obtained by a 4-hour simulation on a Pentium M, using long-run averages with confidence intervals of 95%.

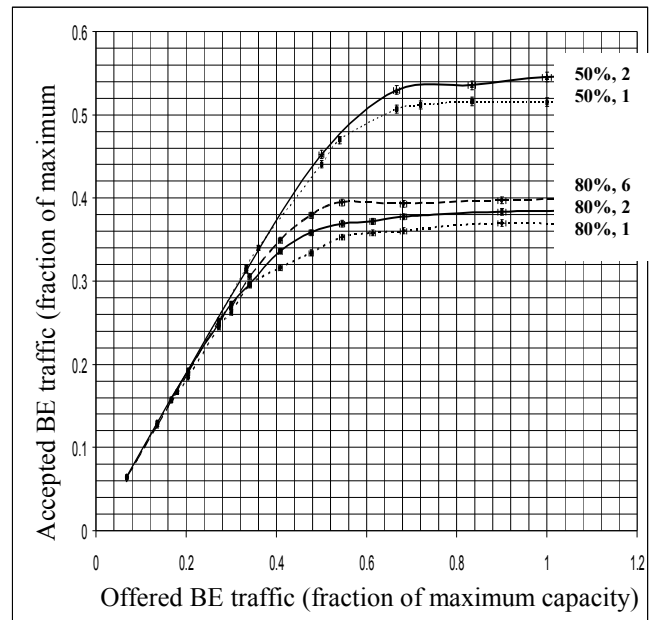
Figure 3 plots average BE packet latency against offered traffic for two GT loads (80% and 50% of capacity), for different path diversities. Note that all path diversities between 1 and 6 were simulated, but only the curves that give significant improvements are shown. Each curve corresponds to the GT load level and path diversity number mentioned on top of it.



**Figure 3: Average BE latency for different path diversities**

Figure 4 plots the realized throughput (accepted traffic) against the offered traffic, i.e. traffic that attempted to be injected into the network. Again, only significant curves are shown. The curves shown correspond to those in Figure 3, i.e. from bottom to top we see GT load 80%, path diversities 1, 2 and 6, and then GT load of 50% with path diversities 1 and 2. The saturation throughput values correspond to those of Figure 3. The network is stable even in very high loads. Naturally, improvements on average throughput (injected traffic) only become apparent towards network saturation.

Table 1 analyzes selected data points from the figures, giving relative improvements of using multi-path compared to the single-path approach. In all cases, a higher saturation throughput is observed. Latency is also shown for selected offered traffic levels. We observe that using a two-path approach gives a typical improvement of 20-30%. In very high GT loads, a six-path split reduces latency by a further 20%! For each path diversity value, the absolute improvement is obviously more marked in the high BE loads, while the relative improvement is steady.



**Figure 4: Accepted versus offered traffic**

**Table 1 Selected data from Figures 3 and 4**

GT load	$P_D$	Sat. Thpt.	Latency for selected BE traffic levels			
			0.1	0.2	0.3	0.35
1	1	0.36	20 cycles	40	96	158
1	2	0.38 (+6%)	16 (-20%)	30 (-25%)	72 (-25%)	112 (-29%)
1	6	0.40 (+17%)	12 (-40%)	24 (-40%)	58 (-40%)	82 (-48%)
			<b>0.2</b>	<b>0.3</b>	<b>0.4</b>	<b>0.51</b>
0.6	1	0.52	22	23	48	114
0.6	2	0.56 (+8%)	12 (-46%)	17 (-26%)	32 (-33%)	84 (-26%)

Using a single path per GT flow chokes the links belonging to this path and has an adverse affect on BE performance through backpressure.

Increased path diversity has two counteracting effects. On one hand it spreads the BW requirement over more links, thus lowering the average load per link. Since BE routing is single-path and oblivious, and BE traffic bursty, a highly congested link delays many BE flits. On the other hand, it increases the interruption of flows that would otherwise be unaffected. Thus while the interruption to some flows is significantly lower, *more* flows are affected. There will therefore be a path diversity value corresponding to the minimum average latency and maximum saturation throughput.

For very high GT loads, we have reached significant improvements with up to 6 paths. Perhaps considering non-minimal paths would improve performance further. For the more realistic GT load, still injecting flits 60% of the time, the optimum path diversity seems to be  $P_D=2$ , and no further improvement is expected in increasing the path diversity beyond 6.

## V. EVALUATION OF APPROACH

This section gives a discussion of the generality and limitations of the results, and a cost analysis of a possible multi-path implementation.

### A. Generality of Results

Although the experiment demonstrates the advantages of the approach well, it is important to realize its limitations before assuming the results to be general. For example, a particular topology and traffic injection models were used. The GT source and sink were constant, and the only network parameters changed were the GB and GT load. Thus, effects of network parameters such as buffer depth were not investigated. In addition, the experiments took place on a transaction-based simulation tool. Although simulation at this level does not simulate a real implementation, its results are indicative of the potential of the approach for a multitude of NoCs.

### B. Cost Analysis

The multi-path approach we consider is cheap to implement. In particular, we consider a form of minimal oblivious GT routing and achieve good results already when going to two paths. In addition, our approach recommends using a simple BE routing algorithm, which potentially cuts design costs.

Given an existing NoC with BE and GT services (the *base network*), one can consider to implement the approach in hardware or as a transparent software layer. The costs involved are as follows:

1) *Configuration Complexity.* Since more paths must be guaranteed compared to the base network, the static planning and configuration complexity will increase. If  $P$  is the set of possible paths for a particular data flow, then there are  $2^{|P|}-1$  possible configurations to choose from, compared to  $|P|$  possibilities in the single path case. This assumes the path diversity is not limited. If the supported path diversity  $n < |P|$ , then the possible configurations are given by the following formula:

$$\sum_{i=1}^n \binom{|P|}{i}$$

Since optimal flow allocation is NP-hard [20], heuristic approaches are sought in practice and are already implemented in the software layer of the base network. These approaches must be modified to cope with the extra complexity.

2) *Data Sequencing.* In a software approach, a software layer that is transparent to the application code splits the data into several flows with separated guarantees. If the packets arriving on all paths have equal latencies, there is no problem. Assume, however, that we have two flows with different guaranteed latencies. Now we seek to merge the data in the correct order on the receiving side. To solve this, we must delay the lower latency flow by the difference in latencies of the two flows. In practice this may not be trivial.

In hardware, this delay of flows translates to extra buffering in the NI, which is sized by the difference in latency between the two flows. The hardware must also insure the correct sequencing of data. In case two paths are used, the receiver can infer the correct sequencing by the port of arrival and

timing by a simple logical function. In case more than two paths are used then extra bits must be inserted to indicate which *path* it traversed. To distinguish between  $N$  paths, the number of bits is given by  $\lceil \log_2 N \rceil$ . The costs therefore depend on the number of paths provided, and can be bounded by the maximum allowed distance between source and destination. This will cost extra logic at the routers.

3) *Extra costs of a hardware approach.* If a hardware approach is chosen, then the hardware must define and store different paths for the same data flow. In simple routing schemes, the method can be implemented cheaply using several passes of intermediate nodes. In the case of source routing storing intermediate nodes is trivial. In the case of simple E-cube routing for example, the flit header could increase in size to contain intermediate destinations. If some GT slack is allowed then a smarter scheme could be devised. For example, a flit could pass through several network interfaces, each calculating the next intermediate destination based on a certain policy (such as a random selection of the minimum paths).

### C. Example: *Æthereal*

Let us qualitatively discuss the costs likely to be incurred when the approach is implemented on an existing platform. In the Philips *Æthereal* network interface [25] for example, each source buffer is associated with a single source buffer and a single path. Alternative paths can be implemented per connection by allowing to store several paths in the Master network interface. The bigger challenge is to prevent a sequencing problem at the slave interface.

In the current flow control mechanism [25] the free space remaining at the destination buffer is recorded, and credits are returned from the destination when packets are consumed. Using this flow mechanism, flow switching would have to stop one flow and wait for all remaining credits to return before starting the next alternative flow. An alternative solution to the packet sequencing problem would be inserting a path identifier to the flit header. The appropriate sequencing can then be inferred from the path ID and the arrival order of the packets (since the injection policy is known).

In addition, we estimate a larger delay in the scheduler, due the increased number of GT paths to be calculated. The scheduler does not need to be modified, however, as the alternative flit paths can be regarded as new paths with known BW requirements.

Because source routing is used, there is no need to store and interpret intermediate destinations.

Thus we conclude that on *Æthereal*, as an example, the method can be implemented with simple hardware modifications of the existing running system, alongside increased static GT schedule complexity. Because of the switching overhead mentioned above, more experimentation will be necessary to investigate the relationship between the frequency of switching and per-flow performance, as well as the performance per flow.

## VI. CONCLUSIONS

In this paper we explored the effect of increasing path diversity of GT traffic on BE performance in a NoC. In particular, we considered latency and throughput for the case of oblivious E-cube BE routing and minimal GT paths on a 4x4 mesh. We showed on a simulation platform that using two instead of one GT path improves average BE latency by 20-30% and increases the saturation throughput by 5-10%. For a reasonable GT load of 60% we reach improvements of 40-50% in latency and 17% in throughput when path diversity is increased to 6. These results provide very promising indication of the potential of our approach.

We find that under reasonable GT loads, increasing path diversity to two already achieves almost maximum improvement in BE performance. This makes the implementation of multi-path routing very cheap. Multi-path routing can be made transparent for the application code, whether using a software or hardware implementation.

In addition, current dynamic optimizations, and adaptive routing mechanisms can be used on top of our approach.

### A. Future Work

Our first next step would be to implementing the approach in reality, i.e. on a real NoC using GT traffic generated by a real multimedia application.

Regarding further simulations, the model can easily be extended to higher-radix topologies, while increasing the number of GT flows and explicitly introducing locality in BE traffic. Since it has been shown that e-cube routing generally performs better than other oblivious and adaptive algorithms under *neighbor traffic* [20]. So we still expect an improvement in results.

Working with non-minimal paths would also be interesting to investigate in the higher GT loads, for which we have not reached the best performance in this paper.

It would also be interesting to investigate the effect of changing different network parameters such as buffer depths at the NI.

An interesting application of the approach is *under-reservation* of links. The essence of under-reservation lies in the fact that the link bandwidth is configured for some bandwidth below worst-case. This increases the bandwidth available for other GT flows, which may also be under-reserved. Suppose that two demand 'peaks' arrive at the link. Then an alternative path, called an *escape path*, could be temporarily used for one of the links. This would prevent the streams from interfering and allow both demands to be met. This would only work if statistically demand peaks almost never arrive at the same time.

## REFERENCES

- [1] Keutzer, Malik, Richard Newton, Rabaey, Sangiovanni-Vincentelli: "System-level design: orthogonalization of concerns and platform-based design". *IEEE Transactions on Computer-Aided Design of Circuits and Systems*, 2000, Volume 19
- [2] SgROI, Sheets, Mihal, Keutzer, Malik, Rabaey, Sangiovanni-Vincentelli: "Addressing the system-on-a-chip interconnect woes through communication-based design". *Proc. DAC*, June 2001, p667
- [3] Bainbridge, Furber: "CHAIN: A delay-insensitive chip area interconnect", *IEEE Micro*, 2002, 22, p16
- [4] Benini, De Micheli: "Powering networks on chips". *Proc. ISSS*, October 2001, p33
- [5] Guerrier, Greiner: "A Generic Architecture for On-Chip Packet-Switched Interconnection". *Proc. DATE*, March 2000
- [6] Dally, Towles: "Route packets, not wires: on-chip interconnection networks". *Proc. DAC*, June 2001, p684
- [7] Benini, De Micheli: "Networks on chips: A new SoC paradigm". *IEEE Comput.*, 2002, 35, p80
- [8] Wielage, Goossens: "Networks on Silicon: Blessing or Nightmare?" *Proc. of Euromicro Symposium on Digital System Design. Architectures, Methods and Tools*, pp. 196-200, 2002
- [9] Goossens, van Meerbergen, Peeters, Wielage: "Networks on silicon: Combining best-effort and guaranteed services". *Proc. DATE 2002*, pp. 423-425
- [10] Rijpkema, Goossens, Radulescu, Dielissen, van Meerbergen, Wielage, Waterlander: "Trade Offs in the Design of a Router with Both Guaranteed and Best-Effort Services For Networks on Chip", *Proc. DATE*, March 2003
- [11] Goossens, Dielissen, van Meerbergen, Poplavko, Rădulescu, Rijpkema, Waterlander, Wielage: "Guaranteeing the quality of services in networks on chip". *Networks On Chip*, Kluwer 2003, ISBN 1-4020-7392-5
- [12] Liu, Zheng, Tenhunen, "Interconnect intellectual property for Network-on-Chip (NoC)", *Journal of Systems Architecture special issue on Network on Chip*, vol. 50, February 2004
- [13] Dielissen, Radulescu, Goossens, Rijpkema: "Concepts and implementation of the Philips Network-on-Chip". Philips Research Laboratories, 2004
- [14] Bolotin, Cidon, Ginosar, Kolodny: "QNoC: QoS architecture and design process for Network on Chip". *The Journal of Systems Architecture*, December 2003
- [15] Bolotin, Cidon, Ginosar, Kolodny: "Cost considerations in networks on chip". *The VLSI Journal*, 2003
- [16] Bolotin, Morgenshtein, Cidon, Ginosar, Kolodny: "Automatic Hardware-Efficient SoC Integration by QoS Network on Chip". *Proc. ICECS*, 2004
- [17] Kumar, Jantsch, Soininen, Forsell, Millberg, Öberg, Tiensyrjä, Hemani: "A Network-on-Chip Architecture and Design Methodology". *Proc. of IEEE Comp. Society*, April 2002
- [18] Millberg, Nilsson, Thid, Jantsch: "Guaranteed bandwidth using looped containers in temporally disjoint networks within the Nostrum network on chip", *Proc. DATE*, February 2004
- [19] Andreasson, Kumar, "On improving Best-effort throughput by better utilization of Guaranteed-Throughput channels in an on-chip communication system", *Proc. IEEE Norchip*, November 2004.
- [20] Dally, Towles: "Principles and Practices of interconnection design", Elsevier 2004, ISBN 0-12-200751-4
- [21] SHESim modeling tool: <http://www.ics.ele.tue.nl/~mgeilen/shesim/shesim.html>
- [22] Ferguson, Huston: "Quality of Service: Delivering QoS on the Internet and in corporate networks", John Wiley & Sons 1998, ISBN 0-471-24358-2.
- [23] Nguyen, Zakhor: "Distributed Video Flowing with Forward Error Connection", 2002
- [24] Murali, De Micheli, "Bandwidth-constrained mapping of cores onto NoC architectures", *Proc. DATE*, February 2004
- [25] Radulescu et al, "An Efficient On-Chip Network Interface Offering Guaranteed Services, Shared-Memory Abstraction, and Flexible Network Configuration.