Efficient Link Capacity and QoS Design for Network-on-Chip

Zvika Guz[1], Isask'har Walter[1], Evgeny Bolotin[1], Israel Cidon[2], Ran Ginosar[2], Avinoam Kolodny[2]
Electrical Engineering Department, Technion, Haifa, Israel
[1]{zguz, zigi, bolotin}@tx.technion.ac.il [2]{cidon, ran, kolodny}@ee.technion.ac.il

Abstract

This paper addresses the allocation of link capacities in the automated design process of a network-on-chip based system. Communication resource costs are minimized under Quality-of-Service timing constraints.

First, we introduce a novel analytical delay model for virtual channeled wormhole networks with non-uniform link capacities that eliminates costly simulations at the inner-loop of the optimization process. Second, we present an efficient capacity allocation algorithm that assigns link capacities such that packet delays requirements for each flow are satisfied. We demonstrate the benefit of capacity allocation for a typical system on chip, where the traffic is heterogeneous and delay requirements may largely vary, in comparison with the standard approach which assumes uniform-capacity links.

1. Introduction

As the density of VLSI designs increases, the number of modules in a typical System on Chip (SoC) is expected to be significantly larger than today. Traditional interconnection schemes might no longer be adequate for these systems. The concept of *Network on a Chip* (NoC) (introduced in [1], [2]) suggests that different modules would be connected by a simple network of shared links and routers. As was analyzed in [3], NoC power and area costs scale much better than shared bus, segmented bus or dedicated point to point connections.

NoC is customized for a particular SoC through an automatic network design phase (e.g., [4], [5], [6]). The goal of the link capacity design process is to minimize the network cost (in terms of area and power) while maintaining the required quality of service (QoS), defined in terms of acceptable packet delays for the specific system communication demands [4]. This customization is based on adequate bandwidth allocation to each of the network links: insufficient allocation will not meet performance requirements while lavish allocation will result in excessive power and area consumptions.

Wormhole routing [7] is an increasingly common interconnect scheme for NoC as it minimizes communication latencies, requires small buffer space and is simple to implement. However, performance evaluation and customization process of wormhole based NoCs heavily rely on simulations as no existing analysis (e.g., [8]-[15]) accounts for the combination of heterogeneous traffic patterns and virtual channels. Unfortunately, these are both fundamental characteristics of NoC interconnect.

The use of simulations makes the task of searching for efficient capacity allocation computationally extensive and does not scale well with the size of the problem. On the other hand, a detailed exact analytical solution of a complex NoC is intractable and simplistic approximations may lead to inaccurate results.

In this work, we propose a hybrid methodology to the network design problem that combines the best of both worlds. First, we propose a novel and simple analysis for a wormhole based NoC, which approximates the network behavior under a wide range of loads. Given any system (in terms of topology, routing and link capacities) and its communication demands (in terms of packets length and generation rate), the analysis estimates the delay experienced by every source-destination pair. To the best of our knowledge, this is the first analysis of a wormhole network with non-uniform link capacities. Next, we suggest a simple algorithm that applies the delay analysis to efficiently allocate capacities to network links, avoiding repetitive simulations at the inner-loop of the optimization process [4]. Simulation is only used for final verification and fine tuning of the system. Using a design example we demonstrate that our algorithm considerably decreases the total NoC cost and significantly improves the speed of the customization process.

2. NoC architecture

Our NoC reference architecture, QNoC [4], is a wormhole based network which guarantees quality of

This work was partially supported by the Semiconductor Research Corporation (SRC), Intel Corp., and the iSRC consortium.

service (QoS) of different inter-module traffic types. QNoC is based on a two dimensional grid topology of irregular mesh and wormhole routing. In wormhole networks, each packet is divided into a sequence of *flit*s which are transmitted over physical links one by one in pipeline fashion. A hop-to-hop credit mechanism assures that a flit is transmitted only when the receiving port has free space in its input buffer. QNoC is lossless, and packets traverse the network on a shortest path using a deadlock free XY routing. Another important feature of NoC is the ability to construct an asymmetric network with variable speed inter-router links. Different network links, connecting routers to other routers or modules, may have different bandwidths set during the design process to meet the QoS requirements.

High performance wormhole based interconnect systems often include *virtual channels* (VCs) which increase NoC throughput [16]. Furthermore, virtual channels must be included when links have different capacities to allow the multiplexing of several slow streams over a high bandwidth link.

Flits of different VCs that contend for the same link bandwidth are time-multiplexed according to some arbitration policy. QNoC employs a simple policy in which flits of the active outgoing VCs are transmitted in a round-robin manner over the physical link. Previous work [16] has showed that adding VCs increases maximal network throughput. As silicon cost of VC is moderate, we assume that physical links are split into an adequate number of VCs. In particular, we assume that a head flit acquires a channel instantaneously on every link it traverses.

3. Wormhole delay model

In a wormhole network, the transit time of a packet between a specific source-destination pair is composed of two components [15]: the time it takes the head flit to reach the destination module (path acquisition time) and the time it takes the rest of the packet to exit the network (transfer time); Path acquisition time is affected by the complex interaction among different flows in the system and transfer time is affected by other flows sharing the same links (link's capacity is time multiplexed among all virtual channels sharing the link).

Since NoC delay/cost tradeoffs are different from those of off-chip networks and since performance is a key issue, it is clear that NoCs should be designed to operate under a relatively light load. Consequently, for the sake of simplicity and computational efficiency, our analysis addresses low to medium loads and does not attempt to achieve high accuracy under very high utilizations.

Our analysis focuses on the transfer of long packets, i.e. packets which are composed of a number of flits significantly larger than the number of buffers along their path. From the simulations presented in [4], it is clear that such packets (termed the *block transfer* class of service) are the ones that place the most stringent demand on NoC resources. However, our analysis can be easily extended to handle multiple classes of service.

We consider a wormhole deadlock free fixed routing network that is composed of N routers connected by unidirectional links. The packets composing each source-destination pair traffic identify a *flow*.

Our model uses the following assumptions:

1. Each flow generates fixed length packets using a Poisson process. (bursty traffic can be modeled by using artificially larger packet size.)
2. Sources have infinite queues and sinks immediately consume flits arriving at their destination.
3. Routers have a single flit input queue per VC.
4. The propagation delay of flits through links and routers is negligible.
5. Back pressure credit signal is instantaneous.

3.1. Notation

The network is characterized using the following notations:

- C_j = *capacity of link j [bits per second]*
- l = *flit size [bits]*
- F = *the set of all flows, from every source module* $1 \le s \le N$ *to every destination module* $1 \le d \le N$
- f^i = *A flow from the set* F
- m^i = *the mean packet length of flow* $f^i \in F$ *[flits]*
- λ^i = *average packet generation rate of flow* $f^i \in F$ *[packet/sec]*
- π^i = *the set of links composing the path of flow* f^i
- π^i_j = *the set of links that are subsequent to link j on flow i's path (a suffix of the path* π^i *)*

The following notations are used to analyze packets' delay:

- T^i = *the mean transit time of packets of flow* f^i *(the average time elapsed since a packet is created until its last flit exits the network)*
- t^i_j = *the mean time to deliver a flit of flow i over link j (waiting for transmission and transmission times)*
- Λ^i_j = *the total flit injection rate of all flows sharing link j except flow* f^i *[flits/sec].*

3.2. Wormhole analysis

The delivery of a packet in a wormhole network resembles a pipeline traversal. When the number of parts composing the packet is considerably larger than the number of pipeline stages, the latency (the time it takes the first bits to exit the pipe) is insignificant compared to the

total time, which in this case is mostly affected by the pipeline's throughput. Since packets are assumed to be considerably longer than the buffers along their path, and since each head flit instantaneously acquires a virtual channel on every link it arrives at, we ignore path acquisition time and approximate transmission time only.

As in a classic pipeline, the transfer time is dominated by the stage with the smallest service rate. Since flits of different flows are interleaved on links, the approximation of t_j^i should account for the transmission time of other flits on the same link. We use a modification of the basic M/M/1 modeling [17] as an approximation of the flit interleaving delay:

$$t_j^i = \frac{1}{\frac{1}{l} \cdot C_j - \Lambda_j^i} \quad (1)$$

where Λ_j^i (defined above) is the bandwidth consumed by all flows other than flow *i* on link *j*. Formally:

$$\Lambda_j^i = \sum_{f \mid j \in \pi^f \wedge f \neq i} \lambda^f \cdot m^f \quad (2)$$

Equation (1) models the mean interleaving delay experienced by flits of flow *i* on link *j* as a simple queue, without accounting for the bandwidth consumed by packets of flow *i* itself. This modification is based on the observation that flits of a flow are interleaved on a physical link due to the delivery of packets that belong to other flows only. By substituting (2) into (1) we get:

$$t_j^i = \frac{l}{C_j - l \cdot \sum_{f \mid j \in \pi^f \wedge f \neq i} \lambda^f \cdot m^f} \quad (3)$$

The total transfer time, which is dominated by the hop with the longest delay, can then be written as:

$$T^i \simeq m^i \cdot \max(t_j^i \mid j \in \pi^i) \quad (4)$$

The above approximation does not capture the inter-link dependencies and is generally too optimistic, for medium and high loads. Wormhole links loads affect each other mainly due to the back-pressure mechanism: a flit must wait for the arrival of a credit for its virtual channel from the following link. Therefore, flit delivery time over a link (t_j^i) is affected by the delivery time in subsequent links of the flow's path. To reflect the effect of flit delay on other links we replace (3) by the following equation which accounts for these inter-link dependencies. Our simulations (Section 5) show that this simple expression successfully estimates the resulting link delay:

$$\tilde{t}_j^i = t_j^i + \sum_{k \mid k \in \pi_j^i} \frac{l \cdot \Lambda_k^i}{C_k} \cdot \frac{t_k^i}{dist^i(j,k)} \quad (5)$$

where $dist^i(k, j)$ is the distance (measured in number of hops) between link *j* and *k* on the path of flow *i*. Formally:

$$dist^i(k, j) = \left| \pi_j^i / \pi_k^i \right| \quad (6)$$

Equation (5) approximates the delay experienced by flits of flow *i* on link *j* by adding to the basic flit delay (t_j^i) a term that takes into account the cumulative effect of the delay of subsequent links along the path. This term is weighted by two factors: the links' distance from link *j* ($dist^i(k, j)$) and the factor by which they are being utilized by flows other than flow *i* itself $(l \cdot \Lambda_k^i / C_k)$. The amendment is based on the observation that the increase in a link's delay is mainly caused by neighboring, congested links.

As explained above, the mean total transit time of each flow is calculated using the longest interleaving delay on its path. Therefore, (4) is replaced by:

$$T^i \simeq m^i \cdot \max(\tilde{t}_j^i \mid j \in \pi^i) \quad (7)$$

In Section 5, we evaluate the quality of our approximated analysis against simulation studies.

4. Capacity allocation

Traditional wormhole networks employ links with uniform capacity set according to the traffic requirements. Since each link is not independently assigned the minimal capacity that is required to meet the end-to-end delay requirement, some links are much faster than needed, consuming unnecessary power and area.

Given a network topology, a routing scheme, communication demands (in terms of packet generation rate and length) and QoS requirements, we suggest an algorithm that adjusts the capacity of each link, striving to minimize the total allocated capacity while assuring that each flow meets its performance requirement.

In order to assign links capacities efficiently, we introduce the following notation:

$T_{REQ}^f =$ *The required mean transit time for flow f*

$\delta =$ *The amount of capacity [bits/sec] added to the network on each iteration*

The capacity assignment minimization problem can now be formalized as follows:

Given:
F
$\forall f \in F: \quad m^f, \lambda^f$
$\forall f \in F: \quad T_{REQ}^f$
$\forall link\ j$, assign link capacities (C_j) s.t.:
$\forall f \in F: \quad T^f \leq T_{REQ}^f$
$\forall link\ j: \quad \sum_{f \mid j \in \pi^f} \lambda^f \cdot m^f \cdot l < C_j$
$\sum C_j$ is minimal

Figure 1: Capacity assignment minimization problem.

Using this formulation, we suggest the algorithm described in Figure 2 to allocate link capacities. The algorithm takes a greedy approach: The initialization phase allocates a minimal preliminary capacity to each link in the network. The main loop analyses each source-destination flow separately. It first uses the delay model (Section 3) to approximate the flow's delay given the current capacity allocation vector (line 5). If the delay is larger than required (line 6), the algorithm allocates a small, predefined amount (δ) of extra capacity to the flow's path: It first assigns the extra capacity to each of the links along the path separately (lines 7-11) and approximates the resulting packet delay. It then searches for the link with the largest sensitivity to bandwidth addition, i.e. the link for which adding capacity results in the shortest delay overall (line 12). The extra capacity is added only to that link (line 13). When the algorithm terminates, all flows meet their required delay.

In practice, the analytical model does not capture all of the complex dependencies and effects in a virtual channeled wormhole network. As a result, the mean delay of a few flows may be under-estimated and the capacity of some links might be too small. Therefore, the resulting assignment is verified using network simulation, and some extra capacity may still be added to these flows' path.

```
      /*assign initial capacities*/
1)   foreach link e:
2)     C_e ← Σ_{f∈F: e∈π^f} λ^f · m^f · l
3)   end foreach

4)   foreach flow f ∈ F :
       /*evaluate current transit delay*/
5)     T^f ← Delay_Model (C, f)
6)     while (T^f > T_REQ^f )
         /*look for most sensitive link*/
7)       foreach e ∈ π^f :
8)         ∀j ≠ e: C̃_j = C_j
9)         C̃_e = C_e + δ
10)        T_e^f ← Delay_Model (C̃, f)
11)      end foreach
12)      e' = argmin_e {T_e^f} /*get most sensitive link*/
13)      C_e' = C_e' + δ     /*increase its capacity*/
14)    end while
15)  end foreach
```

Figure 2: Capacity allocation algorithm.

5. Numerical results

In order to evaluate the delay analysis and to demonstrate the benefit of using the capacity allocation algorithm, we present the following two examples: The first example considers a homogeneous system, in which every module injects the same amount of bandwidth into the network and packets' destinations are chosen randomly with uniform distribution. While this is not typical to SoC, this scenario is very often used to evaluate wormhole network analyses. The second example represents a more realistic traffic for a SoC, exhibiting a non-uniform communication pattern: modules communicate with only a subset of all possible destinations and different flows have different bandwidth and delay requirements. In particular, some modules send and receive packets from a single module (many-to-one and one-to-many patterns), emulating a SoC in which data is generated and destined at specific modules (for example, SoCs with a single main CPU or a single cache memory, or SoCs that use an off-chip DRAM memory). As in typical SoCs, modules that exchange high bandwidth traffic are placed in proximity.

For each example, we compare the analysis with simulation results for a varying utilization factor, by using a wide range of uniform allocation vectors. We then apply the suggested capacity allocation algorithm and present its benefit over uniform link capacities assignment.

We have implemented a tool that automatically assigns link capacities given the network topology, routing scheme, communication demands and QoS requirements. This tool, which uses the aforementioned delay model and capacity allocation algorithm, is to be used by the chip interconnect designer to minimize the network resources. The system was simulated using the OPNET modeler [18]. The model includes all the complex dependencies between different flows in wormhole networks (due to particular virtual channel arbitration schemes, path acquisition time, finite router queues, etc.).

5.1. Homogeneous network example

The network comprises of a regular four by four, two dimensional mesh with XY routing. Links have identical capacities; packets of each flow are generated by a Poisson process, with a mean rate $\lambda = 1/4.8e-4$ [packets/sec]; packets consist of 500 flits, each flit is 16 bit long.

5.1.1. Delay analysis. As can be seen in Figure 3, though all flows inject the same bandwidth, the different distance and different aggregated load of links along their path results in a large diversity in the packet delays. Assuming that all flows have identical requirements, the mean transit delay of some flows is much lower than needed; This slack can be trimmed by a more efficient link capacity scheme.

Figure 4a compares the mean end-to-end packet delay of the analytical model with the simulation results as a function of the most utilized link utilization level (i.e. for a wide range of uniform capacity allocations). The analytical

model closely follows the simulation results for a wide range of loads, way beyond the load that is expected to be found in a practical SoC (the mean absolute error reaches 8% when utilization is over 90%).

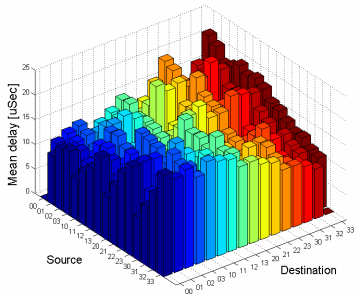

Figure 3: Mean flows delay in the homogeneous system.

x and y axes mark the source and destinations modules respectably and z axis is the mean packet delay (longest delay is 2.5X longer than shortest one).

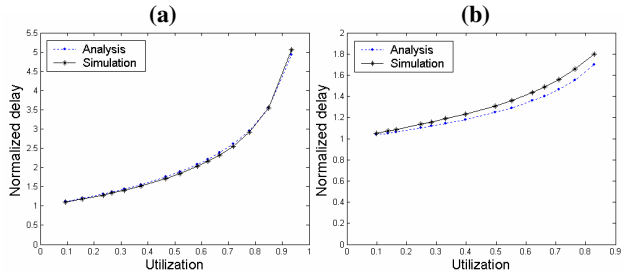

Figure 4: Mean packet delay.

Analytical model estimation and simulations results for the homogeneous system (a) and for the heterogynous system (b). x axis is the utilization of the most loaded link and y axis is the end to end delay normalized by the delay in a zero loaded system (i.e. where no other flows exists).

5.1.2. Capacity allocation results. In order to evaluate the suggested capacity allocation algorithm, we have compared it with uniform capacity allocation for a system with the same packet delay requirements (10uSec for each flow). By using simulations, we found that the total capacity needed in a uniform system is 74.4Gbits/sec (1.55Gb/sec per each one of the 48 inter-router links).

Figure 5a presents the capacities as assigned by the algorithm. A total budget of 69.0Gb/sec was used, which reflects a moderate 7.2% saving of resources. Figure 6 presents the resulting packet delay for each flow. Though less capacity is used, all flows still meet their delay requirement. As expected, the capacity allocation algorithm achieves a modest resources saving in the homogeneous example. Since all source-destination flows exist, it is impossible to reduce the capacity of a large number of links significantly without causing flows to miss their end-to-end delay requirement. Only the capacities of the leftmost links (both upward and downward) were considerably trimmed (e.g. links 00→10, 10→20 and 20→30 in Figure 5a). This is because of their relatively light utilization following the XY routing scheme.

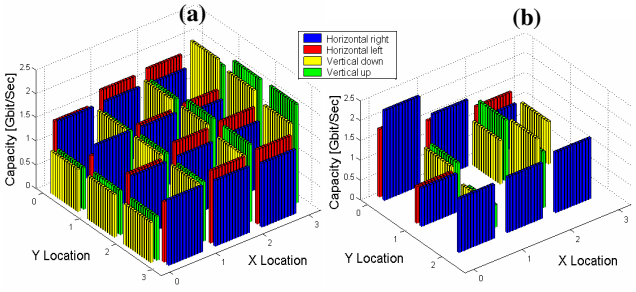

Figure 5: Link capacities.

Link capacities as assigned by the capacity allocation algorithm for the homogeneous system (a) and for the heterogeneous system (b).

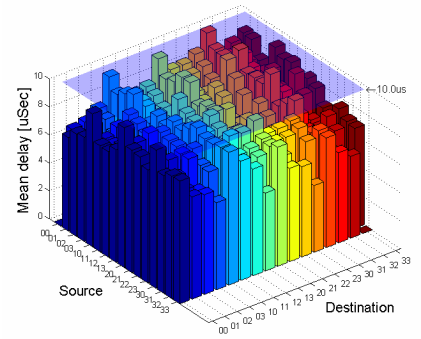

Figure 6: Mean flows delay.

Delay in the homogeneous system when capacities are assigned by the capacity allocation algorithm.

5.2. Heterogeneous network example

The heterogeneous network comprises of a regular three by four, two dimensional mesh with XY routing. Table 1 describes the flows' characteristics and delay requirements.

Flow	Inter-arrival time [uSec]	Packet length [flits]	Total BW [Mb/sec]	Required delay [uSec]
01→12; 01→10; 03→01; 10→01; 12→01; 20→03	66.67	500	120	10
01→00; 12→22	66.67	500	120	5
22→01; 23→13	33.30	500	240	10
01→13; 21→01	500	500	16	10
00→01	16.67	500	480	5
01→02	16.67	500	480	10
01→21	5000	500	1.6	15

Table 1: Heterogeneous system

5.2.1. Delay analysis. Figure 4b compares the mean end-to-end packet delay of the analytical model with simulation results as a function of the most utilized link utilization factor. As can be seen from the figure, the model properly approximates the resulting delays in the

low to medium utilization range. Since NoCs are not expected to operate at high utilization (Section 3), the higher error rates in this region do not present a problem.

5.2.2. Capacity allocation results. Similarly to subsection 5.1.2., we have compared the total budget assigned by the capacity allocation algorithm with a uniform assignment that meets the same requirements. Figure 5b presents the link capacities as assigned by the algorithm. While the uniform assignment consumes a total of 41.8Gb/sec (counting only links with non-zero flow), the algorithm used only 28.7Gb/sec, achieving a reduction of 30% in resources.

Figure 7 presents the mean packet delay as approximated by the model and as evaluated by simulation. The figure also presents the maximal mean packet delay requirement for each flow. Though simple, the analytical model closely approximates the resulting packet delay and all flows meet their delay requirement.

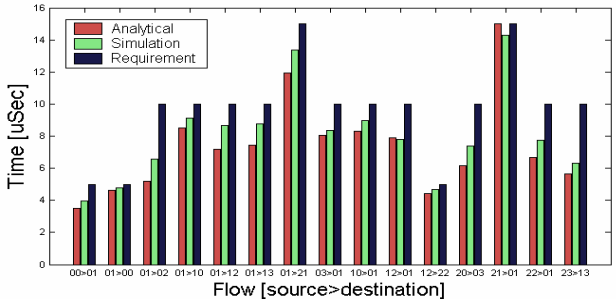

Figure 7: Capacity allocation results.
Mean packet delay as approximated by the model and as extracted from simulation vs. the maximal mean packet delay requirement.

In this example, the capacity allocation algorithm was able to achieve a significant resource saving due to different bandwidth and requirements of the different flows, and due to the diversity in link loads. Since some flows have weaker requirements than others, it is possible to allocate relatively low capacity to some of the links, without causing any flow to violate its delay requirement. As a result, heterogeneous systems are more likely for a substantial resource saving than homogeneous ones.

6. Summary

Allocating different capacities to different links is an important phase in the design process of NoC-based systems. A good assignment algorithm would allocate network resources efficiently so that QoS and performance requirements are met but total cost is minimized.

The paper includes two novel contributions: The first is a simple static timing analysis model that captures virtual channelled wormhole networks with different link capacities and eliminates the reliance on simulations for timing estimations. The paper also suggests an allocation algorithm that greedily assigns link capacities using the analytical model so that packets of each flow arrive within the required time. Using a design example, we showed the potential benefit of automated link capacity allocation in a typical NoC-based SoC, where the traffic is heterogeneous and critical delay requirements vary significantly.

7. References

[1] P. Guerrier and A. Greiner. A Generic Architecture for On-Chip Packet-Switched Interconnections. Proc. DATE 2000

[2] W. J. Dally and B. Towles. Route Packets, Not Wires: On-Chip Interconnection Networks. Proc. DAC 2001

[3] E. Bolotin, I. Cidon, R. Ginosar and A. Kolodny. Cost Considerations in Network on Chip. Journal of Systems Architecture, special issue on Network on Chip, Volume 50, February 2004, pp. 105-128

[4] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny. QNoC: QoS Architecture and Design Process for Network on Chip. Special issue on Networks on Chip, The Journal of Systems Architecture, February 2004

[5] K. Goossens, J. Dielissen, O. P. Gangwal, S. G. Pestana, A. Radulescu and E. Rijpkema. A Design Flow for Application-Specific Networks on Chip with Guaranteed Performance to Accelerate SOC Design and Verification. Proc. DATE 2005

[6] J. Hu and R. Marculescu. Application-Specific Buffer Space Allocation for Networks-on-Chip Router Design. Proc. ICCAD 2004

[7] W.J. Dally and C. Seitz. The Torus Routing Chip. Distributed Computing, vol. 1, no. 3, 1986, pp. 187-196

[8] H. Sarbazi-Azad, A. Khonsari and M. Ould-khaoua. Performance Analysis of Deterministic Routing in Wormhole k-ary n-cubes with Virtual Channels. Journal of Interconnection Networks, 2002, vol. 3

[9] S. Loucif and M. Ould-khaoua. Modeling Latency in Deterministic Wormhole Routed Hypercube under Hot-Spot Traffic. The Journal of Supercomputing, March 2004

[10] C. Roche, P. Palnati and M. Gerla. Performance of Congestion Control Mechanisms in Wormhole Routing Networks. IEEE Infocom'97, Japan, 1997

[11] J. Kim, and C. R. Das. Hypercube Communication Delay with Wormhole Routing. IEEE Trans. on Comp., 1994

[12] R. I. Greenberg and L. Guan. Modeling and Comparison of Wormhole Routed Mesh and Torus Networks. IASTED, 1997

[13] B. Ciciani, M. Colajanni and C. Paolucci. Performance Evaluation of Deterministic Wormhole Routing in k-ary n-cubes. Journal of Parallel Computing, Dec. 1998

[14] J. T. Draper and J. Ghosh. A Comprehensive Analytical Model for Wormhole Routing in Multicomputer Systems. Journal of Parallel and Distributed Computing, 1994

[15] W. Dally. Performance Analysis of k-ary n-cube Interconnection Networks. IEEE Trans. on Comp., June 1990

[16] W. Dally. Virtual Channels Flow Control. Proc. ISCA, 1990

[17] L. Kleinrock. Queuing Systems, volume 1: Theory, John Wiley & Sons Inc, New York, 1975

[18] OPNET modeler (www.opnet.com)