Delay Analysis of Wormhole Based Heterogeneous NoC

Yaniv Ben-Itzhak[1] Israel Cidon[2] Avinoam Kolodny[2]
Electrical Engineering Department
Technion – Israel Institute of Technology
Haifa, Israel
[1]yanivbi@tx.technion.ac.il [2]{cidon, kolodny}@ee.technion.ac.il

ABSTRACT

We introduce a novel evaluation methodology to analyze the delay of a wormhole routing based NoC with variable link capacities and a variable number of virtual channels per link. This methodology can be utilized to analyze different heterogeneous NoC architectures and traffic scenarios for which no analysis framework has been developed before. In particular, it can replace computationally-extensive simulations at the inner-loop of the link capacities and virtual channels allocation steps of the NoC topology optimization process. Our analysis introduces a set of implicit equations which can be efficiently solved iteratively. We demonstrate the accuracy of our approximation by comparing the analysis results to a simulation model for several use-cases and synthetic examples. In addition, we compare the analysis with simulation results for a chip-multi-processor (CMP) using SPLASH-2 and PARSEC traces for both homogeneous and heterogeneous NoC configurations.

Categories and Subject Descriptors

G.1.2 [**Numerical Analysis**]: Approximation – *Nonlinear approximation.*

General Terms

Performance.

Keywords

Networks-on-Chip, Heterogeneous NoC, Analysis-Methodology, Delay Evaluation.

1. INTRODUCTION

NoCs are designed to support a variety of SoC designs with bandwidth and latency requirements for heterogeneous module-to-module flows. In many cases, the SoC communication characteristics are known at design time through specifications that describe the data rates and timing restrictions of each communicating pair. The NoC design parameters and topology are then tailored to meet the given communication requirements, spending minimum power and area. The NoC design process for such systems heavily relies on extensive performance simulations,

where each intermediate NoC configuration is tested against the requirements in a long 'change and test' search sequence. The use of detailed simulations makes the task of searching for efficient link capacities and virtual channels allocation computationally intensive and it does not scale well with the size of the problem. The use of approximated analysis of the NoC behavior replacing simulations can dramatically save time and resources during design. Costly simulations can be left for the final verification and fine-tuning of the system.

This paper explores a novel delay analysis methodology for heterogeneous wormhole based NoCs, with a variable number of virtual channels per link and variable link capacities. The average end-to-end latency per flow is analyzed by calculating its three components: (1) The time it takes the head-flit to leave the source queue (*queuing time at the source*); (2) The time it takes the head-flit to reach the destination module (*path acquisition time*) and (3) The time it takes the rest of the packet to leave the network (*transfer time*).

SoC and CMPs are heterogeneous in terms of module-to-module traffic requirements. Therefore, the appropriate NoC to support such SoCs should be non-uniform in terms of link capacities and virtual channels. However, exiting analysis methodologies [2, 5, 15, 16, 19, 10, 9, 13] are based on the assumption that NoCs are homogeneous. Figure 1 illustrates the delay evaluation of a heterogeneous NoC for two existing extreme analysis methodologies: single-VC [2, 5, 15, 16] and "infinite-VCs", i.e. assuming that virtual channels are always available for any flow [6, 12, 1]. Single-VC based methodologies can result in higher latencies due to the over-estimation of path acquisition latencies (i.e. acquiring VCs along the entire path).

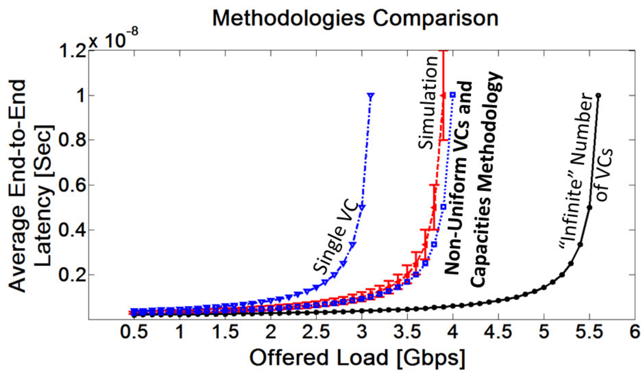

Figure 1. Comparison of different methodology approaches for a heterogeneous NoC. Single-VC ([2, 5, 15, 16]), "infinite" number of VCs, i.e. assuming that VCs are always available for any flow, ([6, 12, 1]) and our new proposed heterogeneous NoC analysis methodology.

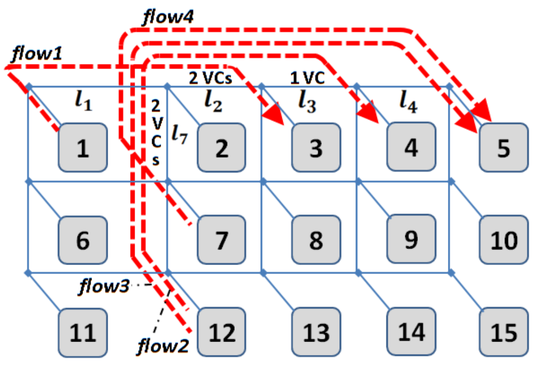

Figure 2. Example for a NoC with non-uniform numbers of VCs (Unless noted, links have sufficient number of VCs).

"Infinite-VCs" based methodologies are also imprecise for two reasons. First, there is a wrong assumption that the path acquisition latencies are instantaneous which under-estimate realistic path acquisition latencies. Second, it is assumed that all potential flows are concurrently multiplexed over each link; this results in higher transfer latencies. Therefore, these two methodologies are not capable of evaluating delay properly. Hence, an analysis methodology for heterogeneous NoCs is required.

Previous works have presented analysis methodologies for NoCs with a single-VC [2, 5, 15, 16], a uniform (same for all links) number of VCs [19, 10, 9, 13] and "infinite" number of VCs [6, 12, 1].

Huang et al. in [8] presented the only previously existing analysis methodology for non-uniform numbers of VCs. However, their analysis accounts for Head-of-line blocking probability and therefore is incapable of calculating many performance metrics such as end-to-end latency. It analyzes the traffic and estimates the congestion on each channel independently rather than having a global view of all the flows by taking into account the mutual interaction between them. While we cannot directly compare our end-to-end latency analysis to this work, our experience and results show that one must consider the mutual interaction of intersecting flows in order to get a reasonable delay approximation. Guz et al. in [6] presented an analysis methodology for non-uniform capacities of links. However, they assume "infinite" number of VCs.

To our knowledge, no existing analysis accounts for the combination of heterogeneous traffic patterns, non-uniform link capacities, and variable numbers of VCs. In the sequel, we illustrate some of the phenomena which can occur in a NoC with non-uniform links including a variable numbers of VCs and variable link capacities, and we address the challenges facing the development of a sound analysis methodology in this case.

Figure 2 depicts a NoC example comprised of links with non-uniform numbers of VCs. There are four flows which can be transmitted over link 2 (l_2). However, since link 7 (l_7) has only two VCs, no more than three flows can simultaneously enter link 2. This phenomenon affects the time it takes for a flow to acquire a VC over link 2; therefore, we should take into account only three waiting flows and not four. We address this phenomenon by introducing the term *aggregate effective number of flows over link X*; it stands for the maximum number of flows that simultaneously can enter any link X. Furthermore, the aggregate interleaving rate experienced by a flow over link 2 depends on the effective number of flows from each ingress link (i.e. links 1 and 7) and their capacities. Therefore, we decompose the *aggregate effective number of flows over link X* into the *effective number of flows from link Y to link X*, where link Y is any ingress link to link X. Section 3.2 describes the definitions and section 4.2 describes how and why to use it.

We present another phenomenon termed *remaining path acquisition time.* Generally, the time for a flow`s head-flit to acquire a VC over a link depends on the time that other interfering flows occupy the link. Clearly, an interfering flow occupies the VC as long as it is transferring; moreover, the interfering flow occupies the VC also during the time it acquires VCs over subsequent links along its path, i.e. the *remaining path acquisition time*. In addition, we take into account that during the *remaining path acquisition time* the interfering flow occupies a VC but is not transmitted; therefore, it is not actually disturbing other transmitting flows. This phenomenon is explained in detail in section 4.2.

Furthermore, we make a distinction between the packet generation rate into the source`s queue and the packet injection rate into the NoC. The injection rates are the values that actually affect the end-to-end NoC latencies and not the values of the generation rates. Therefore, for given generation rates of the sources, our analysis determines the actual injection rates, which can be lower than the generation rates. Section 4.4 describes the procedure for determining the injection rates.

These phenomena also have some relevance for a NoC with uniform number of VCs. However, previous analyses of NoCs with a uniform number of VCs did not take them into account. Therefore, our analysis methodology also offers better approximation for such NoC topologies.

This paper is organized as follows: Section 2 presents the assumptions of our analysis. In section 3 we described the notation, and formally define the terms: *aggregate effective number of flows over link X* and *effective number of flows from link Y to link X*. In section 4 we present details of our novel delay evaluation analysis. Section 5 presents numerical validation results of our analysis. We evaluate several use-cases, synthetic examples and modeling of chip-multi-processor (CMP) with a single shared cache.

2. BASICS OF DELAY ANALYSIS

In a wormhole routing network, the end-to-end latency (T^f) of a packet of flow f, sent between a specific source-destination pair, is the sum of the *queuing time at the source* (q^f), the *path acquisition time* (a^f) and the *transfer time* (t^f). Formally,

$$T^f = q^f + a^f + t^f. \qquad (1)$$

The transfer time, t^f, is affected by other flows sharing the same links, since each link capacity is divided among all active VCs sharing the link. The path acquisition time, a^f, is affected by an even more complex interaction among the flows, as a packet may wait for the evacuation of VCs by other packets sharing links with it, which in turn wait for the evacuation of VCs in other links.

For simplicity, in this paper we use the assumptions that each source generates a single flow and the propagation delay of flits through routers is negligible. However, the analysis can be easily extended for sources with multiple flows and for routers with non-zero latency.

Other assumptions are: all flows generate packets according to a Poisson process; sources have infinite packet queues and destinations immediately consume arriving flits; routers have a single flit input queue per VC; the wormhole back-pressure credit signal is instantaneous; and the routing algorithm is deterministic.

3. NOTATIONS AND DEFINITIONS

3.1 General Notations and Definitions

We denote each unidirectional link by l_j ($j \in \{1, \dots, |L|\}$, where $|L|$ is the total number of links in the NoC). Link i (l_i) can be either a unidirectional link between network routers or a unidirectional access link that connects the module to the router.

We define $d(l_i, l_j)$ as the minimal number of routers connecting links i and j.

Definition: $In(l_j) \triangleq \{l_i | d(l_i, l_j) = 1\}$.

$In(l_j)$ is defined as the group of all ingress links to l_j (link j). The network is defined using the parameters listed in Table 1.

Table 1. Parameter Definitions

$flit$	Flit size [bits].
m^f	The mean packet length of flow f [flits].
λ^f	Packet generation rate of flow f [packets/sec].
$\lambda^f_{network}$	Packet injection rate of flow f into the network [packets/sec].
C_{l_j}	Capacity of link j [bits/sec].
V_{l_j}	Number of VCs on link j.
π^f	Ordered set of consecutive links that compose the path of flow f.
$\pi^f_{l_j}$	Set of subsequent links to link j over the path of flow f, i.e. a suffix of the path π^f.
F_{l_j}	The group of flows sharing link j, $F_{l_j} \triangleq \{f | l_j \in \pi^f\}$.
F_{l_i,l_j}	The group of flows sharing link j from the ingress link i. i.e.: $F_{l_i,l_j} \triangleq \{f | l_i \in \pi^f, l_j \in \pi_{l_i}{}^f, d(l_i, l_j) = 1\}$.
$F^{Eff}_{l_i,l_j}$	*Effective number of flows from link i to link j.* Defined in sub-section 3.2.
$F^{Eff}_{l_j}$	*Aggregate effective number of flows over link j.* Defined in sub-section 3.2.

The source generates packets of flow f at a rate of λ^f. The packets are queued at the source queue and injected into the network in a FIFO manner. When the source queue is instable, the packet injection rate of packets into the NoC, $\lambda^f_{network}$ is lower than the packet generation rate, λ^f, i.e. $\lambda^f > \lambda^f_{network}$. Clearly, $\lambda^f_{network}$ is the value that affects the network NoC latencies (i.e. path acquisition time plus transfer time) and not the value of λ^f. In section 4.4 we describe an iterative procedure to evaluate $\lambda^f_{network}$ values for given values of λ^f.

3.2 Effective Number of Flows

A non-uniform numbers of VCs over the NoC can cause different phenomena as illustrated by Figure 2. We take them into account by using the following notations in our analysis:

$F^{Eff}_{l_i,l_j}$ – *Effective number of flows from link i to link j.*

The maximum number of flows that simultaneously can enter link j from the ingress link i.

$F^{Eff}_{l_j}$ – *Aggregate effective number of flows over link j.*

The maximum number of flows that can simultaneously enter link j from all ingress links. It is equal to the sum of $F^{Eff}_{l_i,l_j}$ over all ingress links.

Figure 2 presents an example for $F^{Eff}_{l_i,l_j}$, $F^{Eff}_{l_j}$, F_{l_i,l_j} and F_{l_j} . For instance, since l_7 has only two VCs (i.e. $V_{l_7} = 2$), the maximum number of flows that can enter simultaneously from l_7 to l_2, $F^{Eff}_{l_7,l_2}$, is equal to two. However, there are three flows (numbered 2,3 and 4) that can be transmitted from l_7 to l_2, denoted by the group F_{l_7,l_2}. Similarly, since l_2 has only two VCs (i.e. $V_{l_2} = 2$), $F^{Eff}_{l_3}$ is equal to two, and $F^{Eff}_{l_4}$ is equal to one because l_3 has only a single VC.

We formally define $F^{Eff}_{l_i,l_j}$ and $F^{Eff}_{l_j}$ using the recursive procedure described in Appendix A. For a given link j and ingress link i, the procedure finds, for each flow from link i to link j, F_{l_i,l_j}, the link with the minimum number of VCs over its path till link j. Since these links can limit the number of flows that simultaneously can enter link j, we sum up the number of VCs of those links. Finally, $F^{Eff}_{l_i,l_j}$ is equal to the minimum between this value, the number of flows from link i to link j, V_{l_i} and $F^{Eff}_{l_i}$. After $F^{Eff}_{l_i,l_j}$ is calculated for all ingress links to link j, $\{In(l_j)\}$, $F^{Eff}_{l_j}$ can also be calculated.

4. END-TO-END LATENCY ESTIMATION

The analysis methodology approximates the average end-to-end latency of flow f, T^f, which is composed of the *queuing time at source*, q^f, the *path-acquisition time*, a^f, and the *transfer time*, t^f (1).

In order to calculate a^f, we add up the acquisition time of every link j along flow f`s path, $a^f_{l_j}$. Therefore,

$$a^f = \sum_{l_j \epsilon \pi^f} a^f_{l_j} . \qquad (2)$$

The transfer time, t^f, is dominated by the hop with the smallest bandwidth available to flow f along its path, and is therefore computed as following:

$$t^f = m^f \cdot \max\left\{t^f_{l_j} \middle| l_j \in \pi^f\right\}. \qquad (3)$$

Where $t^f_{l_j}$ is the flit transfer time of flow f over link j.

Note: For very short packets (i.e. when the packet length is much shorter than the number of buffers along its path), the approximation of (3) is not sufficient. We need to calculate the latency of the head-flit apart from the latency of the body-flits. Thus, alternatively, the following equation can be used:

$$t^f = \sum_{l_j \epsilon \pi^f} t^f_{l_j} + (m^f - 1) \cdot \max\left\{t^f_{l_j} \middle| l_j \in \pi^f\right\}. \qquad (4)$$

Finally, by substituting (2) and (3) into (1) we get the average end-to-end latency of a packet of flow f,

$$T^f = q^f + \sum_{l_j \epsilon \pi^f} a^f_{l_j} + m^f \cdot \max\left\{t^f_{l_j} \middle| l_j \in \pi^f\right\}. \quad (5)$$

In the following sub-sections, implicit equations are derived for q^f (4.1), $a^f_{l_j}$ (4.2) and $t^f_{l_j}$ (4.3). The *aggregate effective number of flows over link X*, the *effective number of flows from link Y to link X* and the *remaining path acquisition time* phenomena result in dependency between the equations. These equations are based on heuristics and approximations. Finally, we solve by iterations the set of implicit equations in order to evaluate the average end-to-end latency of each flow. The variables and their corresponding equations are listed in Table 2.

Table 2. Variable Definitions

q^f	Queueing time at the source of flow *f* [sec]	Equation (6)
a^f	Total path acquisition time of flow *f* [sec]	Equation (2)
$a^f_{l_j}$	Path acquisition time of flow *f* over link *j* [sec]	Equations (7), (8), (9), (10)
t^f	Total transfer time of flow *f* [sec]	Equation (3) or (4) (Depends on the packet length)
$t^f_{l_j}$	Flit transfer time of flow *f* over link *j* [sec]	Equations (11), (12), (13), (14)

4.1 Queuing Time at the Source

We approximate the source queuing time using the M/D/1 queue model [11]:

$$q^f = \frac{t^f + a^f}{2}\left(\frac{(t^f + a^f)\cdot\lambda^f}{1-(t^f + a^f)\cdot\lambda^f}\right). \quad (6)$$

Clearly, when a flow does not share any of its links with other flows, (6) is the exact mean queuing time, since the time required to deliver a packet through the network $(t^f + a^f)$ is deterministic. When a packet might be multiplexed with other packets within the network, the service time is not deterministic any more. However, thorough simulations [6] show that (6) is a good approximation for the queuing time even for flows that are frequently multiplexed.

4.2 Path Acquisition Time over a Link

The path-acquisition time of flow *f* over link *j*, $a^f_{l_j}$, is the time for the head-flit of the packet to acquire a VC over the link. Clearly, path-acquisition is instantaneous whenever the number of VCs of the link exceeds the number of flows that traverse this link. This section addresses the case when there is possible competition among the flows for the same VCs. In such a case, the head-flits of different packets acquire VCs in FIFO manner.

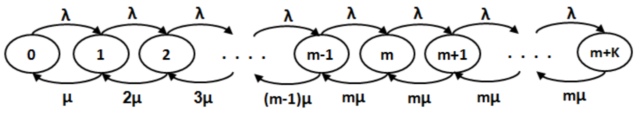

Figure 3. M/M/m/K queue model.

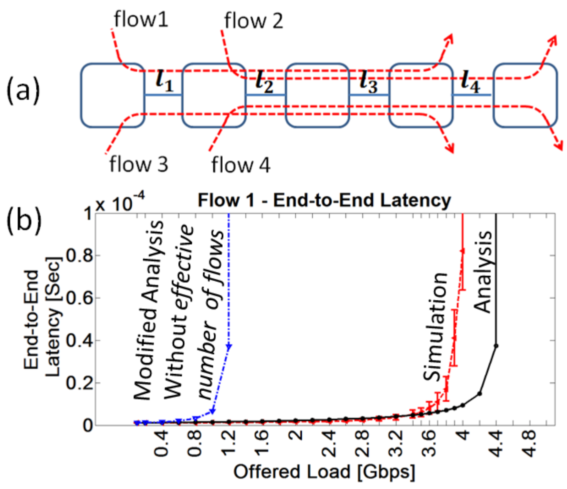

Figure 4. Rationale for using *effective number of flows* (An example).

$$a^f_{l_j} = \begin{cases} 0 & F^{Eff}_{l_j} \le V_{l_j} \\ \text{M/M/m/K queue} & F^{Eff}_{l_j} > V_{l_j} \end{cases}. \quad (7)$$

We use $F^{Eff}_{l_j}$(*aggregate effective number of flows over link j*) in order to determine whether there are enough VCs to avoid competition (7), rather than the total number of flows over link *j*, $\left|F_{l_j}\right|$.Our numerical examples and comparisons to simulations show that this results in a much better numerical accuracy. Figure 4 presents an example for such a case. All the flows have the same packet generation rate, and all the links have the same capacity and a single VC (see Figure 4(a)). Figure 4(b) presents the end-to-end latency of flow 1 reported by simulation, by the analysis and by a modified analysis which substitutes $F^{Eff}_{l_j}$ with $\left|F_{l_j}\right|$. For instance, the modified analysis uses the number of all flows over l_3, i.e. $\left|F_{l_3}\right|$, instead of $F^{Eff}_{l_3}$ in order to estimated path acquisition time of flow 1`s over l_3. At first glance it seems that there are four possible flows which can acquire a VC over l_3 ($F_{l_3} = \{1,2,3,4\}$). However, since flow 1 already "won" over the previous links there is only one possible flow which can acquire this VC ($F^{Eff}_{l_3} = 1$) and it is flow 1 itself; therefore, the path acquisition of flow 1 over l_3 is instantaneous.

Assume that a packet head-flit of flow *f* arrives to link *j*. The head-flit should wait until all previously queued head-flits acquire a VC. Therefore, we approximate the time to acquire a VC using the M/M/m/K (m-Servers, Finite storage) queue model (Figure 3). The queuing time is an approximation for the time it takes for all previously queued head-flits to acquire a VC. Therefore, the queue consists of V_{l_j} servers (i.e. $m = V_{l_j}$) and $F^{Eff}_{l_j} - 1$ waiting positions (i.e. $K = F^{Eff}_{l_j} - 1$). The arrival rate is equal to the sum of arrival rates of all flows that share link *j* with flow *f*, i.e.:

$$\lambda = \sum_{k\epsilon\{F_{l_j}\setminus f\}} \lambda^k_{network}\,. \quad (8)$$

The service rate is expressed by (9); it is equal to the average rate that the flows group F_{l_j} occupies the VCs. The rate that flow *k* occupy a VC is equal to the inverse of its transfer time, t^k, plus the path acquisition time along its subsequent path to the destination, $\sum_{l_i \epsilon \pi_{l_j}^k} a_{l_i}^k$ (*remaining path acquisition time*).

$$\mu = \frac{\sum_{k \epsilon F_{l_j}} \left(\frac{1}{t^k + \sum_{l_i \epsilon \pi_{l_j}^k} a_{l_i}^k} \right)}{\left| F_{l_j} \right|} \qquad (9)$$

Figure 5 illustrates the rationale for adding the *remaining path acquisition time* expression. Assume that all the flows have the same packet generation rate, and all the links have the same capacity and a single VC (see Figure 5(a)). Figure 5(b) presents the end-to-end latency versus the packet generation rate of the flows for flow 2 reported by the simulation, the analysis and a modified analysis that does not take into account the *remaining path acquisition time*. It can be seen that ignoring this delay results in gross inaccuracy. The path acquisition time of flow 2 depends on the time that flow 1 occupies l_1; this time is equal to its *remaining path acquisition time* (i.e. acquiring-VC over l_2, $l_3$and l_4: $a_{l_2}^1$+$a_{l_3}^1$+$a_{l_4}^1$) plus its transfer time (t^1).

$$a_{l_j}^f = \frac{Average\ Queue\ Size}{Average\ Arrival\ Rate} = \frac{\sum_{k=m+1}^{m+K} (k-m) P_k}{(1 - \mathrm{P}_{\mathrm{m+K}})\ \lambda}. \qquad (10)$$

Finally, using Little's law [11] we calculate the path-acquisition time, $a_{l_j}^f$ (10).Where $\{P_k\}$ are the equilibrium probabilities of the M/M/m/K queue model (Figure 3).

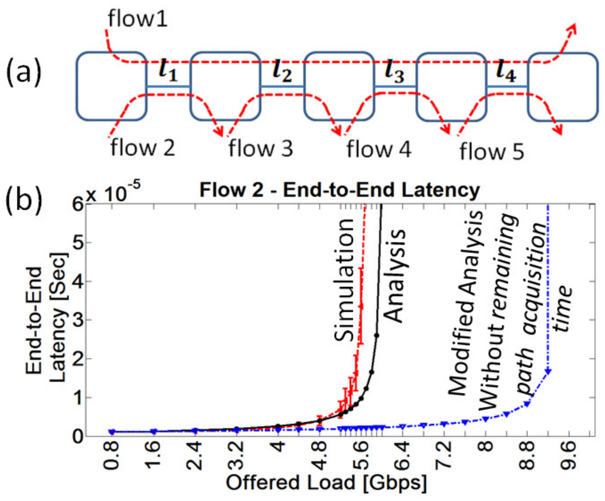

Figure 5. Rationale for using *remaining path acquisition time* (An example)

4.3 Flit Transfer Time over a Link

Since flits from different flows are multiplexed over a link, the flit transfer time of flow *f* over link j, $t_{l_j}^f$, should account for the flit-transmission of other flows on the same link. $t_{l_j}^f$ is calculated by modifying the basic M/M/1 service rate [11] to account for the flows multiplexing:

$$t_{l_j}^f = \frac{1}{\frac{1}{flit} C_{l_j} - B_{l_j}^f}. \qquad (11)$$

Where $B_{l_j}^f$ [flits/sec] is the effective bandwidth consumed by all flows other than flow *f* on link *j*.

$$B_{l_j}^f = \begin{cases} \frac{\min(V_{l_j}-1, F_{l_j}^{Eff}-1)}{F_{l_j}^{Eff}-1} \sum_{l_i \in In(l_j)} B_{l_i,l_j}^f & ; F_{l_j}^{Eff} > 1 \\ 0 & ; F_{l_j}^{Eff} = 1 \end{cases} \qquad (12)$$

B_{l_i,l_j}^f [flits/sec] is the effective bandwidth consumed by all flows other than flow *f* on link *j* from the ingress link i.

$$\mathrm{bw} \triangleq \sum_{k \in \{F_{l_i,l_j} \setminus f\}} \left(\frac{t^k}{t^k + \sum_{l_i \epsilon \pi_{l_j}^k} a_{l_i}^k} \right) \lambda_{network}^k \cdot m^k \qquad (13)$$

$$B_{l_i,l_j}^f = \begin{cases} \mathrm{bw} & ; F_{l_i,l_j}^{Eff} = \left| F_{l_i,l_j} \right| \\ \frac{F_{l_i,l_j}^{Eff}}{\left| F_{l_i,l_j} \right|} \mathrm{bw} & ; l_i \notin \pi^f \ \&\ F_{l_i,l_j}^{Eff} < \left| F_{l_i,l_j} \right| \\ \frac{F_{l_i,l_j}^{Eff}-1}{\left| F_{l_i,l_j} \setminus f \right|} \mathrm{bw} & ; l_i \in \pi^f \ \&\ F_{l_i,l_j}^{Eff} < \left| F_{l_i,l_j} \right| \end{cases} \qquad (14)$$

Equation (13) addresses the effective bandwidth of all the multiplexed flows from the ingress link i. Each such flow is not transmitted over link *j* while it still acquires VCs along its path to destination (i.e. during the *remaining path acquisition time*). Therefore, the packet generation rate of the flow is multiplied by the ratio $t^k / \left(t^k + \sum_{l_i \epsilon \pi_{l_j}^k} a_{l_i}^k \right)$. Furthermore, B_{l_i,l_j}^f depends on whether all multiplexed flows from the ingress link i can be transmitted simultaneously on link j or not. For the first case, B_{l_i,l_j}^f is equal to the sum of the bandwidths over all the multiplexed flows. For the latter case, the effective bandwidth is lower since not all the flows can be transmitted over link j. Therefore, B_{l_i,l_j}^f is decreased by the ratio of $F_{l_i,l_j}^{Eff} / \left| F_{l_i,l_j} \right|$. Hence, (14) results in the effective bandwidth of the multiplexed flows while taking into account the lack of VCs over previous links to link i of the multiplexed flows. Finally, since flow *f* is already being transmitted and consequently occupies a VC, we use (12) in order to bound the total number of interleaved flows over link *j* by $V_{l_j} - 1$.

The total transfer time of flow *f*, t^f, which is dominated by the hop with the smallest per-flow rate, is calculated by (3). As mentioned above, for very short packets (4) can be used instead.

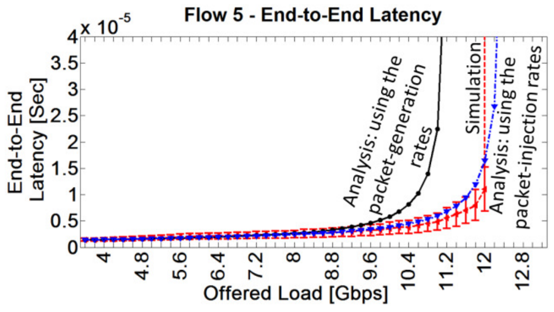

Figure 6. The accuracy improvement of the iterative procedure. (For end-to-end latency of flow 5 depicted in Figure 5(a)).

4.4 Iterative Procedure for Finding $\lambda_{network}^f$

The analysis consists of a set of implicit equations for the variables, q^f, $a_{l_j}^f$, $t_{l_j}^f$, that can easily be solved (see Table 2). Thereafter, we evaluate the end-to-end latency, T^f.As mentioned above, the actual packet injection rate to the NoC could be lower than the packet generation rate. Therefore, using the given packet generation rate of flow f, λ^f, may cause inaccurate results. In order to obtain the packet injection rate to the network, $\lambda_{network}^f$, we first solve the equation set using the λ^f values (i.e. $\lambda_{network}^f = \lambda^f$). Thereafter, for every instable source queue, we use a lower $\lambda_{network}^f$ value and solve the equations again. We repeat this procedure in an iterative manner until we achieve the minimal $\lambda_{network}^f$ that still causes instability at the source queue for each proper flow.

Figure 6 illustrates the accuracy improvement of the iterative procedure for flow 5 depicted in Figure 5(a). It presents the end-to-end latency when taking into account the packet generation rates and the packet injection rates obtained by the procedure. It can be seen that using the packet injection rates significantly improves the accuracy.

5. NUMERICAL RESULTS

The analysis was programmed using Matlab. For a given NoC architecture (i.e. capacity and number of VCs of each link, topology and routing) we generate the equations (see Table 2) and solve them using a standard Matlab non-linear solver. Then, we compare the analysis results to an event-driven (flit-level) NoC simulator written in OMNeT++ [17]. The implemented simulator supports any heterogeneous NoC configuration in terms of every link capacity and number of VCs. The simulator executes wormhole switching with VCs employing round-robin arbitration and deterministic XY routing. We simulate an asynchronous and ideal router (i.e. no internal latency for the NoC router) with a single flit input-buffer for each VC.

5.1 Evaluation of Use Cases

In order to evaluate our analysis, we use a 4x4 2D mesh NoC with a single VC for all links. We simulate a uniform traffic pattern with the same packet generation rate for all the sources. Then, we evaluate the end-to-end latency of the flow over the diagonal path of the NoC (i.e. from the left-bottom tile to the right-top tile, $N_{1,4} \rightarrow N_{4,1}$). Figure 7 presents the end-to-end latency of this flow reported by simulation, our analysis and the analysis presented in [6]. It can be seen that our analysis results in a better approximation. The inaccuracy of our analysis is less than 8% for offered load less than 5 Gbps; the inaccuracy of the saturation threshold is less than 4%. Moreover, using the "infinite-VCs" assumption results in inaccurate analysis [6].

In addition, we compare against the simulation the results of our analysis for a multimedia application (MMS) introduced in [7]. This multimedia application used for the evaluation of end-to-end latency for the methods proposed in [16] and [5]. We manually map the application into 4x4 2D mesh NoC with a single VC for all links. Figure 8 presents the average end-to-end latency as reported by the simulation and our analysis. The difference between our analysis and the simulation is less than 2%; moreover, our analysis accurately computes the saturation threshold. The same accuracy is also achieved for a NoC with two VCs for all links.

5.2 Synthetic Example

In this section, we present a synthetic example of heterogeneous NoC (see Figure 9); all flows have the same packet generation

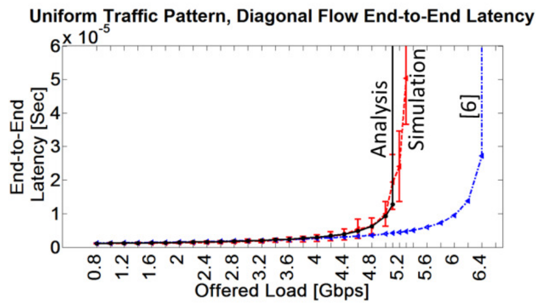

Figure 7. End-to-end latency of the diagonal flow $N_{1,4} \rightarrow N_{4,1}$ for 4x4 NoC with uniform traffic pattern, reported by: simulation, our analysis and the "infinite-VCs" analysis presented in [6].

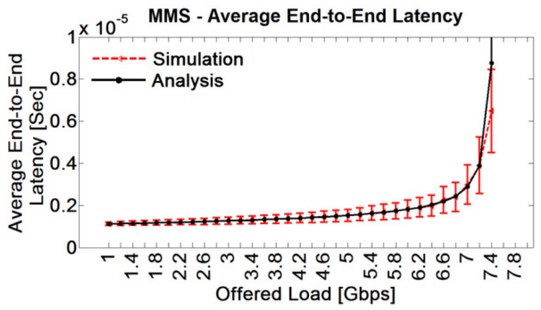

Figure 8. Average end-to-end latency of multimedia application (MMS) [7], reported by simulation and our analysis.

rate. Figure 10 presents the end-to-end latency of each flow. As can be seen, our analysis offers good approximation; the inaccuracy of the saturation threshold is less than 4% for flow 1 and less than 2% for flows 2, 3 and 4.

5.3 Application and Benchmark-Based Comparison

We evaluate the results of our analysis with trace-based CMP traffic. We model a chip multi-processor (CMP) with a single shared-cache (the cache line size is 256B) which acts as hot-module. The traces of the benchmarks are produced using Simics simulator [14] running SPLASH-2 [18] and PARSEC [4] benchmarks. Then, we simulate the CMP using OMNeT++, where each module generates packets according to a given trace.

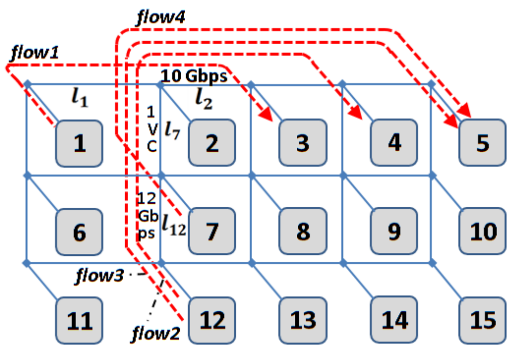

Figure 9. Synthetic example of NoC with non-uniform numbers of VCs and non-uniform capacities of links (Unless noted links are 16Gbps and have sufficient number of VCs). $C_{l_2} = 10\ Gbps; C_{l_{12}} = 12 Gbps; V_{l_7} = 1$.

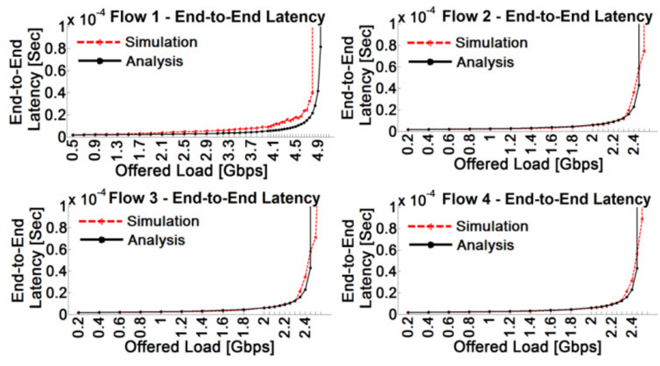

Figure 10. The end-to-end latency of the flows in Figure 9.

We evaluate the average end-to-end latency using our analysis. Furthermore, we compare our analysis to the one presented in [6] which assumes an "infinite" number of VCs.

Figure 11(a) presents the results of the simulation, our analysis and the analysis presented in [6] for several benchmarks. First, we consider a homogenous NoC topology (i.e. all links have the same capacity and number of VCs). The results are averaged for several homogenous NoC topologies. The evaluation of our analysis provides a good end-to-end latency approximation. Moreover, it is much more accurate than the results of [6].

Figure 11(b) presents the average end-to-end latency results for a heterogeneous NoC topology. The results are averaged for several heterogeneous NoC topologies. Our analysis offers more accurate results than [6]. Moreover, the accuracy differences between our analysis and [6] is higher in comparison with the homogenous topologies.

In addition, we model the SoC applications presented in [3] for homogenous and heterogeneous NoC topologies. The applications are manually mapped into the NoC. Figure 12 presents the average end-to-end latency results of the simulation, our analysis and the analysis presented in [6]. The results are averaged for several homogenous and heterogeneous NoC topologies respectively. As can be seen, our analysis accurately computes the end-to-end latency.

5.4 Run-Time Comparison

In this section, we demonstrate the run-time saving gained by using our analysis compared to the simulation. The run-time is measured for simulation executed with 15 cores and analysis executed with a single core (i.e. the analysis uses 6.6% of the resources in comparison with the simulation).

Table 3 presents the simulation run-time and our analysis run-time for several cases presented along the paper. Our analysis offers significant time and computing-resource saving (99.9% and 93.4% respectively).

6. CONCLUSION

A novel delay evaluation analysis to calculate the average end-to-end latencies per flow of a heterogeneous NoC with variable link capacities and number of VCs per link has been presented. Several crucial phenomena in a heterogeneous NoC were observed: the *aggregate effective number of flows over link X*, the *effective number of flows from link Y to link X* and the *remaining path acquisition time*. The quality of our approximation was improved by using these observations. Our analysis offers accurate end-to-end latency evaluation for different NoCs architectures and traffic scenarios.

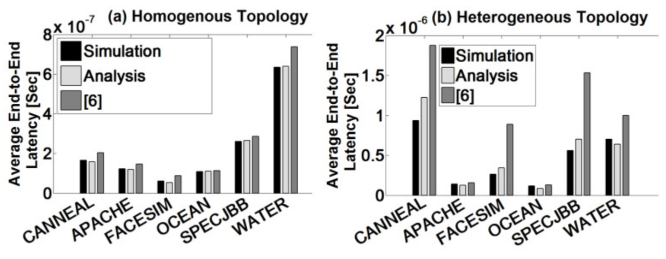

Figure 11. A single shared cache NoC-based CMP with (a) homogenous and (b) heterogeneous topologies. Comparison between simulation, our analysis and the analysis presented in [6] for SPLASH-2 and PARSEC benchmarks.

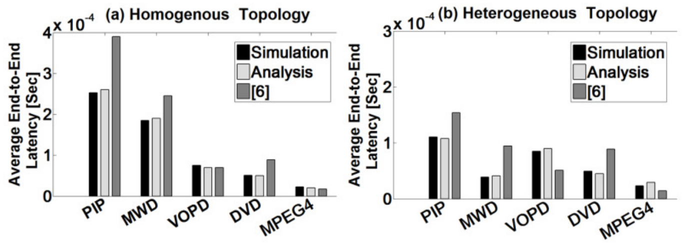

Figure 12. A single shared cache NoC-based CMP with (a) homogenous and (b) heterogeneous topologies. Comparison between simulation, our analysis and the analysis presented in [6] for the applications presented in [3].

Table 3. Run-Time Comparison (For Analysis Computing-Resource Saving of 93.4%)

	Simulation Run-Time [Sec]	Analysis Run-Time [Sec]
Figure 4	3780	4.34
Figure 5	9615	5.4
Uniform traffic pattern (Figure 7)	13725	11.4
MMS (Figure 8)	12585	4.45
Synthetic example (Figure 9)	7455	4.7

7. REFERENCES

[1] M. Bakhouya, S. Suboh, J. Gaber, and T. El-Ghazawi. Analytical modeling and evaluation of on-chip interconnects using network calculus. In *Proceedings of the 2009 3rd ACM/IEEE International Symposium on Networks-on-Chip*, pages 74–79. IEEE Computer Society, 2009.

[2] P. Beekhuizen and J. Resing. Performance analysis of small non-uniform packet switches. *Performance Evaluation*, 66(11):640–659, 2009.

[3] D. Bertozzi, A. Jalabert, S. Murali, R. Tamhankar, S. Stergiou, L. Benini, and G. De Micheli. NoC synthesis flow for customized domain specific multiprocessor systems-on-chip. *IEEE Transactions on Parallel and Distributed Systems*, 16(2):113–129, 2005.

[4] C. Bienia, S. Kumar, J. Singh, and K. Li. The PARSEC benchmark suite: Characterization and architectural implications. In *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*, pages 72–81. ACM, 2008.

[5] S. Foroutan, Y. Thonnart, R. Hersemeule, and A. Jerraya. An analytical method for evaluating Network-on-Chip performance. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2010*, pages 1629–1632. IEEE, 2010.

[6] Z. Guz, I. Walter, E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny. Network delays and link capacities in application-specific wormhole NoCs. *VLSI Design*, 2007, 2007.

[7] J. Hu and R. Marculescu. Energy-and performance-aware mapping for regular NoC architectures. *IEEE Transactions on computer-aided design of integrated circuits and systems*, 24(4):551–562, 2005.

[8] T. Huang, U. Ogras, and R. Marculescu. Virtual channels planning for networks-on-chip. In *Quality Electronic Design, 2007. ISQED'07. 8th International Symposium on*, pages 879–884. IEEE, 2007.

[9] A. Kiasari, D. Rahmati, H. Sarbazi-Azad, and S. Hessabi. A Markovian Performance Model for Networks-on-Chip. In *Parallel, Distributed and Network-Based Processing, 2008. PDP 2008. 16th Euromicro Conference on*, pages 157–164. IEEE, 2008.

[10] J. Kim, D. Park, C. Nicopoulos, N. Vijaykrishnan, and C. Das. Design and analysis of an NoC architecture from performance, reliability and energy perspective. In *Proceedings of the 2005 ACM symposium on Architecture for networking and communications systems*, pages 173–182. ACM, 2005.

[11] L. Kleinrock. Queueing systems, volume 1: theory, 1975.

[12] E. Krimer, M. Erez, I. Keslassy, A. Kolodny, and I. Walter. Packet-level static timing analysis for NoCs. In *Networks-on-Chip, 2009. NoCS 2009. 3rd ACM/IEEE International Symposium on*, page 88. IEEE, 2009.

[13] M. Lai, L. Gao, N. Xiao, and Z. Wang. An accurate and efficient performance analysis approach based on queuing model for network on chip. In *Computer-Aided Design-Digest of Technical Papers, 2009. ICCAD 2009. IEEE/ACM International Conference on*, pages 563–570. IEEE, 2009.

[14] P. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hållberg, J. H\"ogberg, F. Larsson, A. Moestedt, and B. Werner. Simics: A full system simulation platform. *COMPUTER,*, pages 50–58, 2002.

[15] N. Nikitin and J. Cortadella. A performance analytical model for Network-on-Chip with constant service time routers. In *Computer-Aided Design-Digest of Technical Papers, 2009. ICCAD 2009. IEEE/ACM International Conference on*, pages 571–578. IEEE, 2009.

[16] U. Ogras and R. Marculescu. Analytical router modeling for networks-on-chip performance analysis. In *Design, Automation & Test in Europe Conference & Exhibition, 2007. DATE'07*, pages 1–6. IEEE, 2007.

[17] A. Varga et al. The OMNeT++ discrete event simulation system. In *Proceedings of the European Simulation Multiconference (ESM'2001)*, pages 319–324, 2001.

[18] S. Woo, M. Ohara, E. Torrie, J. Singh, and A. Gupta. The SPLASH-2 programs: Characterization and methodological considerations. In *Proceedings of the 22nd annual international symposium on Computer architecture*, pages 24–36. ACM, 1995.

[19] Y. Wu, G. Min, M. Ould-Khaoua, H. Yin, and L. Wang. Analytical modelling of networks in multicomputer systems under bursty and batch arrival traffic. *The Journal of Supercomputing*, 51(2):115–130, 2010.

Appendix A

We define F_{l_i,l_j}^{Eff} and $F_{l_j}^{Eff}$ using the following recursive procedure:

```
Initialization:
Set global array EffFlows to NaN.
$\left(F_{l_i,l_j}^{Eff}, F_{l_j}^{Eff}\right)$= Effective Number of Flows Calc ($l_j$) {
(1) $PrevLinks_{l_j} \triangleq \left\{l_i \middle| l_i \in \pi^f, l_j \in \pi_{l_i}{}^f, d(l_i, l_j) = 1, f \in F_{l_j}\right\}$
(2) for each $l_i \in PrevLinks_{l_j}$ {
(3)   $\left(F_{l_x,l_i}^{Eff}, F_{l_i}^{Eff}\right)$ = Effective Number of Flows Calc($l_i$)
(4)   if ($F_{l_i}^{Eff} > V_{l_i}$)
(5)       for each $k \in \left\{\begin{matrix} f \mid f \in F_{l_i}, EffFlows(f,1) = NaN \\ or\ EffFlows(f,1) \geq V_{l_i} \end{matrix}\right\}$
(6)   $EffFlows(f) = \{V_{l_i}, l_i\}$
(7)     end
(8) end
(9)   $sameBNF = \left\{B_{V_{l_k},l_k} \middle| \begin{matrix} m \in B_{V_{l_k},l_k} \Leftrightarrow m \in F_{l_i} \cap F_{l_j}, \\ EffFlows(m) = \{V_{l_k}, l_k\} \end{matrix}\right\}$
(10)  $F_{l_i,l_j}^{Eff} = \min\left(\sum_{B_{V_{l_k},l_k} \in sameBNF} V_{l_k}, \left|F_{l_i} \cap F_{l_j}\right|, V_{l_i}, F_{l_i}^{Eff}\right)$
(11) end
(12) $F_{l_j}^{Eff} = \sum_{l_i \in In(l_j)} F_{l_i,l_j}^{Eff}$
}
```

The procedure gets l_j and returns F_{l_i,l_j}^{Eff} and $F_{l_j}^{Eff}$. It uses a global array called *EffFlows*. It contains two values for each flow: number of VCs of a link and link identifier, (e.g.,$EffFlows(f,1) = V_{l_k}, EffFlows\,(f,2) = l_k$). Before using the procedure, we initialize *EffFlows* to NaN (i.e. Not a Number).

We find the group of ingress links to link j (l_j) , $\{In(l_j)\}$, (line (1)). For each link i (l_i), we recursively execute the procedure (line (3)). Then, for each of the flows over link $i$we check whether $F_{l_i}^{Eff}$>V_{l_i} and if this link has the minimum number of VCs along the path (until link i). If yes, we set *EffFlows* entries to V_{l_i} and l_i respectively (lines (4)-(8)). We find all groups of flows over both links i and j which have the same *EffFlows* entry, $\{sameBNF\}$ (line (9)). We set F_{l_i,l_j}^{Eff} to the minimum between: sum of VCs entries of *EffFlows* of the $sameBNF$ flows group, number of flows from link i to link j, V_{l_i} and $F_{l_i}^{Eff}$ (line (10)). Finally, we calculates $F_{l_j}^{Eff}$ (line(12)).