A Markov Chain Based Method for NoC End-to-End Latency Evaluation

Sahar Foroutan[1,2], Yvain Thonnart[2], Richard Hersemeule[1], Ahmed Jerraya[2]
[1] ST-Microelectronics, [2] CEA-Leti
{sahar.foroutan, yvain.thonnart, ahmed.jerraya}@cea.fr, richard.hersemeule@st.com

Abstract

This paper presents a generic analytical method to estimate communication latency between a source and a destination of a given Network-on-Chip. This method is based on Markov chain stochastic processes. In order to solve the limiting problem of state-space explosion in complex stochastic processes, we propose to construct a reduced Markov chain model for each node of the path, and to recursively use the local mean latencies to obtain the mean latency of the complete path. Comparison between the analytical results obtained by our method and those of a corresponding SystemC CABA simulation platform shows the accuracy of our method.

1 Introduction

Nowadays due to tight time-to-market constraints the success of a SoC (System-on-Chip) design flow completely relies on its ability to perform fast and exhaustive validation and performance evaluation of the full system in the earliest stages of the design. Since the interconnect platform affects every stages of SoC design, fast and accurate performance evaluation of NoCs (Networks-on-Chip) are being increasingly important; time-consuming simulation-based methods don't match anymore with the increasing system complexity; they are slow, non-exhaustive, and non-scalable with the system size, also they are achieved relatively late in the design flow. Novel mathematical and formal methods for both validation and performance estimation of Systems-on-Chip have seemed to be adequate alternatives for traditional simulation-based methods.

The introduction of Network-on-Chips (NoCs) [1-4] as a new interconnecting solution for communication within complex SoCs has led to the definition of many NoC architectures, implementation strategies, and network performance evaluation methods [5-9]. NoCs with probabilistic communication delays and random time intervals caused by resource sharing, contention, and different traffic patterns can be regarded as systems with stochastic characteristics. In this context various performance issues of NoCs were subject of several stochastic and statistical explorations. For example authors of [10, 11] analyze the expected deadline miss ratio of task graphs by considering stochastic versus fixed or worst case task execution times. Authors of [12] address the problem of link capacity allocation in NoCs through a statistical approach. They approximate the distribution of the load generated by all traffics on each link of the NoC with Gaussian distributions and suggest a capacity allocation algorithm, with predictable performance guarantees, that illustrates how much capacity is needed to service for example 90%, 95% or 100% of all traffics. [13, 14] address traffic models of on-chip communications and present statistical traffic patterns for on-chip communication versus traditional traffic.

This paper addresses the NoC end-to-end communication latency (packet latency). The communication latency is an essential metric in the NoC performance evaluation and provides information to estimate the application runtime. Expressed as a function of the load injected to the network, communication latency can also determine the maximum acceptable load (saturation threshold) of the network. But, the definition of latency varies from one work to another. It can be measured per data word, header, packet, or message transfer (several packets) while including or excluding the time at the source queue. [15] provides useful indications for evaluating the performance of complex systems and explains different definitions of the latency in the literature.

The term packet latency is interpreted in this work as the latency of a *tagged-packet* traversing the NoC form a given source to a given destination. It is counted from the moment the tagged packet header arrives to the input port of the source router until the instant it gets out of the destination router. The basic idea of our method is to compute separately the mean latency of each node of the tagged-packet's path. To do so we construct a reduced Markov chain model for each router of the path, steady state analysis of this model gives the mean latency of the router. Since the latency of a node depends on the latency of its following node and so on, computations of node-latencies must follow the reverse order of dependencies (beginning from the end of dependency backward to its root). We use a recursive algorithm that regarding the NoC topology, routing algorithm and traffic pattern finds all latency dependencies and returns the node latencies according to the proper computation order.

This paper is structured as follows: section 2 reviews related work and describes our contributions. The theory of our approach and the proposed Markov chain model of generic router are presented in section 3. Section 4 explains the latency computation order. In section 5 a refinement technique is presented. Section 5 includes experimental results and the comparison between the analytical and simulation results. Finally section 6 concludes the paper.

2 Related Works

NoC performance evaluation is traditionally based on simulation [16-18], however analytical techniques ,using queuing theory in the most of the time, have been proposed to measure the communication latency of NoCs and parallel computer networks (off-chip communication interconnects)[19-25]. Some of them target only a particular network topology such as [22-24] which are restricted to k-ary n-cubes network topologies. Some others place limitation on buffer or packet size, like [21] which propose an analytical model for wormhole routing delay but restricted to networks with single flit buffer capacity. The method presented in this paper is generic in the sense that it supports arbitrary network topologies and deadlock-free routing algorithms with arbitrary packet length. Concerning the applications, we target known traffics with any repartition on the NoC as long as it can be approximated with Poisson distributions (modeled by mean values).

Authors of [21] aim to approximate the end-to-end transfer delay for application-specific flows with packets longer than the buffers along their path. They ignore link acquisition time by assuming that there are as many virtual channels as the number of flows sharing the same physical link (so every head flit of any flow can acquire a VC instantaneously on every link it traverses). Using queuing models they approximate the link transmission time by accounting the time attributed to other virtual channels (i.e. other flows) which interleave over the same physical link. Actually since it focuses on transmission delays, [21] analyzes a “quasi circuit switching” case while our work investigates packet switching case dominated by both link acquisition delay and link transmission delay.

Authors of [25] present a more general queuing theory based model addressing wormhole routing with arbitrary size messages and finite buffers under application-specific traffic patterns also supporting arbitrary network topology and deterministic routing. While the framework of our approach relies on the same assumptions, our methodology is quite different from theirs: our approach is based on a novel Markov chain model for router and the obtaining of the mean router latency by steady-state analysis of this model, whereas [25] relies on the generalization of the single queue model to multiple queues in the router. Using that approach the backpressure effect due to downstream contention tends to be minimized, and thus the saturation effect arises later than what can be observed in simulations.

3 Modeling the NoC as Markov Chain

The basic idea of this work is to build a generic Continuous Time Markov chain (CTMC) model by considering the probability and delay of contentions between the tagged packet and other disrupting packets at each node of the path. Steady state analysis of such model can give the mean value of the latency between the source and the destination of the path. Constructing a single model for the entire path confronts with state-space explosion and restricts the scalability and the generality of the method. Because even if we stay at a high level of abstraction, the size of Markov chain state-space grows exponentially with the number of routers per path and the number of input/output ports per router. The alternative solution, which enormously reduces the complexity of the Markov chain model and is easily generalizable to any path length and any number of I/O ports per router, is to build separated models for each router of the path and to compute the mean latency of each router and finally to obtained the mean latency of entire path from the sum of router mean latencies.

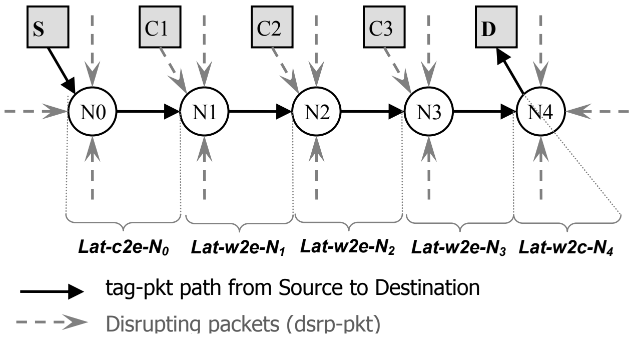

Fig.1: At each node of the path *disrupting packets* appear probabilistically in front of *tag-pkt*, occupy its output ports and increase its latency.

Fig.1 shows a generic path that can belong to any NoC architecture. Solid arrows demonstrate the tagged packet’s path from the source to the destination while dashed arrows show disrupting packets competing with the tagged packet. Packets are considered atomic (i.e. they cannot be divided into shorter pieces by the network) so when the header of a packet arrives to a port the packet body arrives immediately after it. A generic path is composed of n generic nodes which are abstract routers with m input/output ports.

3.1 Router Markov Chain Model

However our proposed model is generic and compatible with routers with any number of input/output ports, but for the reason of simplicity in explanation we focus on routers with five I/O ports. Ports are specified by their direction. Each ***output*** port represents also the link it is connected to. The ports to/from the cores are called core-output and core-input respectively. Supposing that tagged packet has already crossed the path up to node N_p, fig.2 shows the tagged packet waiting at the west input port of this node to obtain the east output port. With some probability it has contention with disrupting packets coming from the north, the south, and the core, waiting for the same output.

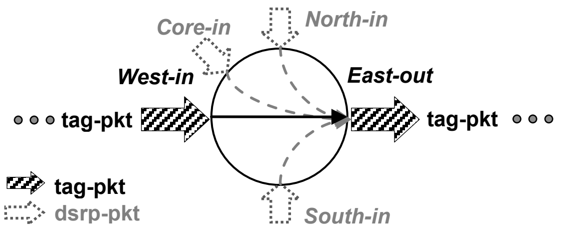

Fig.2: tag-pkt arrives at west input port of N_p and waits for east output port

Each time a disrupting packet is granted before the tagged packet, it causes a delay so the proposed Markov chain is made of probabilistic transitions representing the contention probabilities and also delay transitions representing the delay caused by contentions. Fig.3 demonstrates the Markov model, related to the latency of crossing N_p from the west input to the east output and is abbreviated by "*Lat-w2e-*N_p".

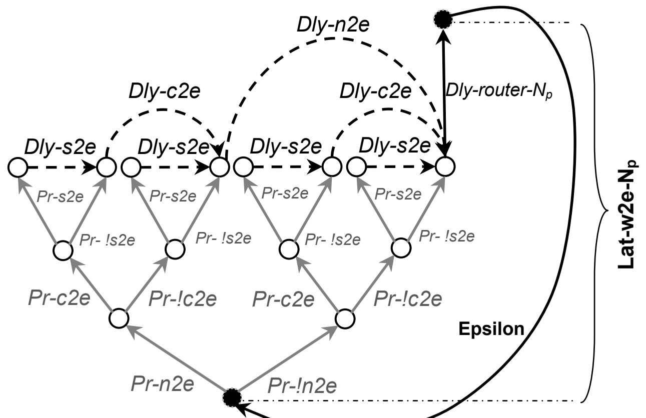

Fig.3: Markov chain underlying the latency of traversing router N_p from the west input to the east output (*Lat-w2e-N_p*)

Contention probabilities are labeled with ***Pr-i2o*** or ***Pr-!i2o***. *Pr-i2o* expresses the probability of contention with a disrupting packet coming from input-port "*i*" going to output-port "*o*". *Pr-!i2o* expresses the reverse. For example "*Pr-n2e*" is the probability of contention with a disrupting packet from north input-port which addresses the east output-port while "*Pr-!n2e*" indicates the inverse and is equal to *1 – Pr-n2e.* Stages of probabilistic transitions perform combining probabilities related to all possible contentions. For instance sequence *Pr-!n2e*➔*Pr-!c2e*➔*Pr-!s2e* depicts the probability in which the east output is free and tagged packet can passes through N_p while *Pr-n2e*➔*Pr-c2e*➔*Pr-!s2e* represents the probability of contention with both north and core input ports.

Delay transitions represent distributions of stochastic delays caused by contentions and are labeled with ***Dly-i2o***. In other words they are the probability distributions of time-durations, during which tagged packet is stalled. Different sequences of delay transitions correspond to different combinations of contention probabilities. For example the worst case sequence *Pr-n2e*➔*Pr-c2e*➔*Pr-s2e* is followed by the sequence *Dly-n2e*➔*Dly-c2e*➔*Dly-s2e* that corresponds to the total time duration in which tagged packet is stalled by three disrupting packets from north, south and core. Delay transition ***Dly-router-N_p*,** indicates the "*router latency*" (the delay needed for traversing router N_p even when the router is empty). Sequences of delay transitions converge to a single state (filled circle) which denotes that the east output-port is allocated to the tagged packet. Later (in sections 2.3 and 2.4) we determine contention probabilities and delays on the basis of NoC architectural parameters and traffic pattern. In this work we consider a fair arbiter mechanism (in the sense that it is symmetric in terms of probability of selection of simultaneous packets on the inputs). Other arbiter mechanism however can be modeled by adding more stages of probabilities.

3.2 Steady-state analysis

While performing a transient analysis of the evolution of the state of a continuous time Markov chain would allow to obtain the probability distribution to reach the final state over the time, it is preferable in terms of computation time, to study only the steady state (at the equilibrium) of the continuous time Markov chain. There are several tools can be used to compute the delays and throughputs of the transitions of a continuous-time Markov chain. We used "BCG_steady" of the "CADP" toolbox [26]. Instead of computing the delay from the initial state to the final state, where the steady state would be the state in which the tagged packet is eventually routed (since the arbitration is fair), we modify the Markov chain to introduce a loop from the final state to the initial state, defining a periodic behaviour. The added transition *"Epsilon"* in our Markov model (fig.3) represents a very short time-duration having no impact on the total time-duration of the model. Its throughput at the equilibrium is equal to the inverse of the mean delay (at equilibrium) from the first state to the last state (the two filled states of fig.3), i.e. the mean latency of traversing router N_p from the west input to the east output.

3.3 Probabilistic transitions

In a given router *N* the probability of contention between the tagged packet coming from ***j*** and a disrupting packet coming from ***i*** competing for output ***o*** is shown with $c_{ij}(o)$ (equivalent to *Pr-i2o* in our Markov model) and is equal to:

$$c_{ij}(o) = P_{io}P_{jo}$$

Where P_{io} is the probability of the presence of data on link ***i*** addressing link ***o***. In our method we assume that the tagged packet is already present so P_{jo} = 1 and the contention probability is equal to P_{io}:

$\boldsymbol{c_{ij}(o) = P_{io}}$ *(at the presence of tagged packet on* ***j****)*

The *forwarding rate* $\boldsymbol{\lambda_{io}}$ (the rate of data traversing the router from input ***i*** and output ***o***) could be a preliminary estimation for P_{io} as presented in equation (*). This approximation is unrefined and in section 4 we present an iterative method to make it more accurate.

$$P_{io} \approx \lambda_{io} = \frac{s_{io}}{s_i} \cdot \lambda_i \qquad (*)$$

Where $\boldsymbol{s_i}$ denotes the number of (source, destination) flows crossing link ***i***, and $\boldsymbol{s_{io}}$ denotes the number of (source, destination) flows crossing both link ***i*** and ***o***. For a deterministic routing algorithm we have:

$s_i = \#\{(p\ ,\ q) \mid R_{pq}(i) = 1\}$

$s_{io} = \#\{(p\ ,\ q) \mid R_{pq}(i) = 1,\ R_{pq}(o) = 1\}$

Where $\boldsymbol{R_{pq}(i)}$ is the indicator function such that $\boldsymbol{R_{pq}(i) = 1}$ if the data sent from the source ***p*** to the destination ***q*** is routed through the link ***i*** and $\boldsymbol{R_{pq}(i) = 0}$ otherwise:

$$R_{pq}(i) = \begin{cases} 1 & \text{if the p to q flow passes through i} \\ 0 & \text{otherwise} \end{cases}$$

In the case of adapting routing algorithms, if we are able to determine statistically the percentage of data sent from each of possible routes between source ***p*** and destination ***q*** then we are able to use the same formalism and $R_{pq}(i)$ will be equal to the ***fraction*** of data from ***p*** to ***q*** transmitted from input port ***i*** to output port ***o***.

λ_i in (*) is the total (average) traffic rate crossing link ***i*** of the NoC and is equal to:

$$\lambda_i = \sum_p \sum_q d_{pq} \cdot R_{pq}(i)$$

Where d_{pq} is the $(p\ ,\ q)^{th}$ element of distribution traffic matrix ***D***:

$$D = \begin{bmatrix} 0 & d_{12} & d_{13} & \dots & d_{1n} \\ d_{21} & 0 & d_{23} & \dots & d_{2n} \\ d_{31} & d_{32} & 0 & \dots & d_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ d_{n1} & d_{n2} & d_{n3} & \dots & 0 \end{bmatrix}$$

Traffic matrix determines the distribution of all traffics (flows) over the NoC. For a NoC with ***n*** nodes the traffic matrix is of size $n*n$ and its $(p\ ,\ q)^{th}$ element (d_{pq}) represents the fraction of data generated by core p which addresses core q.

3.4 Delay transitions

We have two kinds of delay in our Markov model; the router latency (*Dly-router-*N_p) which is the router service time for the header flit and is a function of router design given to the methodology as an input parameter, and the delay caused by disrupting packets (*Dly-i2o-*N_p) which includes two following components:

1) The link transfer delay (*Dly-link-xfr*): the time duration a disrupting packet takes to be transferred through the link $N_p \rightarrow N_{p+1}$ and is a function of both packet and link characteristics such as packet length, link bandwidth, the number of virtual channels sharing same physical link, etc.

2) The acquisition delay (*Dly-Acq-*N_{p+1}): The delay a disrupting packet needs for the acquisition of its output port at the next node. It is counted from the moment the disrupting packet header is at input port of N_{i+1} until the moment it reaches its output port. The computation of this delay depends on the contentions of following nodes. For example in fig.4 the tagged packet is blocked at node N_p by a disrupting packet coming from the north while the later can be stalled at node N_{p+1} waiting for its output port. According to the routing algorithm and the traffic matrix, a disrupting packet having crossed N_p can be routed to north, south, east or core directions of N_{p+1} while each of them may still be occupied by other disrupting packets. So the acquisition of each output-port causes a latency similar to *lat-w2e-*N_p and can be obtained from a Markov model similar to fig.3.

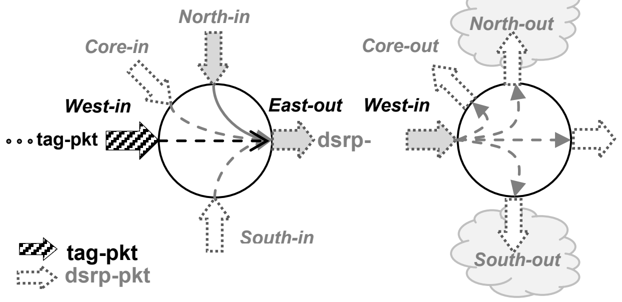

Fig.4 Tag-pkt is waited at N_p by a disrupting packet coming from north which causes a delay according to its output choice at N_{p+1}.

Since a disrupting packet can be routed to different output links of N_{p+1}, *Dly-acq-*N_{p+1} is equal to the average latency needed for acquiring each of them. If ***O*** be the set of all possible outputs of N_{P+1} to which a disrupting packet from ***i*** may be routed then we have:

$$Dly\_Acq\_N_{P+1} = \sum_{o\, \in\, O,\, o \neq i} f_{io} \cdot lat\text{-}i2o\text{-}N_{P+1}$$

Where f_{io}, $\boldsymbol{o} \in O$ is the forwarding probability that determines the probability that a packet coming from input $\boldsymbol{i}$ of N_{P+1} is forwarded to output $\boldsymbol{o}$:

$$f_{io} = \frac{s_{io}}{s_i}$$

For example in the case of fig.4 we have:

$$Dly\text{-}Acq\text{-}N_{p+1} = \sum_{o \in \{N,E,S,C\}} f_{wo} \cdot lat\text{-}w2o\text{-}N_{P+1}$$

The computation of $lat\text{-}w2o\text{-}N_{p+1}$, $o \in \{N,E,S,C\}$ is like $lat\text{-}w2e\text{-}N_p$ and is obtained from a similar Markov chain which depends in the same way on the latencies of the following nodes performing a sequence of dependent latencies. We call it "*latency-dependency sequence*" and we propose a recursive method that first determines the latency-dependency sequence involved with a desired node and then computes the dependent latencies recursively from the tails of the sequence back to the desired node.

4 Latency computation order

In a formal way the latency-dependency sequence is a group of packets that may be blocked waiting on each other to release shared resources (buffers, virtual channels, ports etc.) and making a sequence or a chain. An appropriate formalism for representing the dependency-sequences is the so-called "*Channel Dependency Graph*" (CDGs) [9] which is widely used for detecting low-level deadlocks of routing algorithms.

Since this work addresses deadlock-free routing algorithms there is no cycle in the latency-dependency sequences and they eventually terminate to cores, where there isn't any following node after. Therefore for a given router, the latency between a given input-port and the core output-port (i.e $lat\text{-}i2c\text{-}N_p$) can be obtained independently from a single Markov chain. So by beginning from cores we compute the mean latency of each node of the sequence and use it to compute the mean latency of its backward dependent node and continue until the desired latency. Latency dependency sequences and consecutively the latency computing order are tightly dependent on NoC topology, routing algorithm, and traffic pattern. The following subsection shows how we determine the latency computing order for the example of a 2D-mesh NoC using X-First routing algorithm.

4.1 2D-mesh NoC

Figs 5 and 6 demonstrate the dependency sequences involved with the latencies between the core input port of $N_{3,5}$ and its south and east outputs respectively.

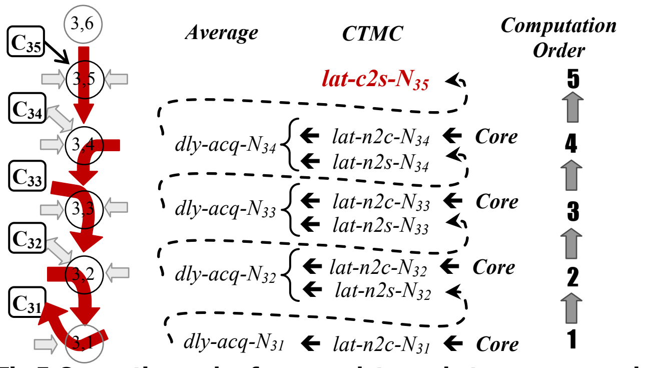

Fig.5 Computing order for mean latency between *core* and *south* ports of *$N_{3,5}$* (*Lat-c2s-$N_{3,5}$*)

The mean latency from the core to the south of $N_{3,5}$ (*lat-c2s-$N_{3,5}$*) is involved with a dependency sequence extending until $N_{3,1}$ (*lat-n2c-$N_{3,1}$*). According to x-first routing, packets are routed first on x axis then on y axis. Disrupting packets (wide arrows in fig.5) competing with the tagged packet for the south output, arrive from north, east or west of $N_{3,5}$ and they address one of the cores $C_{3,4}$, $C_{3,3}$, $C_{3,2}$ or $C_{3,1}$. For example a disrupting packet from north of $N_{3,5}$ occupying the south output could be routed to either core or south ports of $N_{3,4}$, each of them could be blocked by other packets. Therefore both latencies of *lat-n2c-$N_{3,4}$* and *lat-n2s-$N_{3,4}$* are needed for *lat-c2s-$N_{3,5}$*. In the same way *lat-n2s-$N_{3,4}$* depends on *lat-n2c-$N_{3,3}$* and *lat-n2s-$N_{3,3}$* and so on. In the worst case, as shown in fig.5, the sequence of blocked packet may extends until the core $C_{3,1}$ which is the end of dependency (there is no other output choice regarding x-first routing). So the latency computation order begins with *lat-n2c-$N_{3,1}$* and backs up toward $N_{3,5}$.

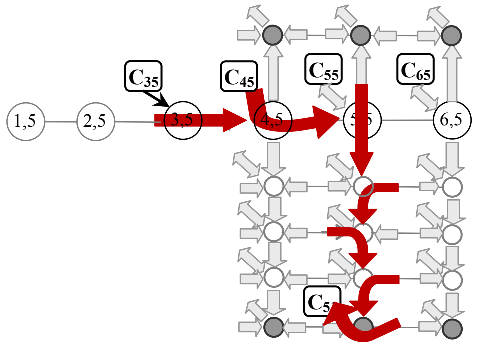

Fig.6 Computing order for mean latency between *core* and *south* ports of *$N_{3,5}$* (*Lat-c2s-$N_{3,5}$*)

In the case *lat-c2e-$N_{3,5}$*, the sequence of blocked packets does not only expand to east direction but also to south and north directions (such as the example shown in fig.6). Latency computing order in this case begins from south and north extremities (filled nodes in fig.6) and from east to west.

5 Iterating Refinement

As mentioned before each time the output port ***o*** at node N_p is blocked by a disrupting packet, the later causes a delay in front of the stalled packet. This delay results in increasing the probability of the presence of data passing from link ***i*** to link ***o***. So, approximating the probability of data presence with the data rate (equation *) is only correct in quasi-zero loads, where the probability of contention is almost zero. To refine this preliminary approximation, we should consider the reciprocal impact of different flows which compete for the same output, on each other. If *lat-i2o-*N_p is the average delay imposed to each packet to traverse node N_p, and L is the packet length, we have:

$$P_{io} = F(Lat\_i2o\_N_p) = \lambda_{io} \frac{L + Lat\_i2o\_N_p}{L}$$

To compute and refine the reciprocal impact of different flows (which address the same output of a router) on each other we perform iterations on all of the competing inputs of that router. In each iteration, one of the competing inputs is marked as the tagged packet input ($\boldsymbol{i_{Tag}}$) and others are marked as disrupting packet inputs. For example in fig.8 four inputs $\boldsymbol{i_1}$, $\boldsymbol{i_2}$, $\boldsymbol{i_3}$ and $\boldsymbol{i_4}$ of router N_p compete for the output ***o.*** Each iteration aims to compute the latency between i_{Tag} and o (*lat-*i_{Tag}*2o-*N_p). So the tagged packet arriving from i_{Tag} is in contention with disrupting packet coming from the other inputs. The latency between i_{Tag} and o (*lat-*i_{Tag}*2o-*N_p) is obtained from our Markov chain model (fig.3), in which the probabilities and delays are taken from the previous iteration.Table.1 shows five iterations corresponding to fig.8. For the first iteration the link transfer delay, caused by each disrupting packet is approximated only with the length of that disrupting packet and we obtain *lat-*i_1*2out-*N_p*(iter_1)*. The later is then used in the second iteration to approximate *Dly-link-xfr* of a disrupting packet coming from i_1 (i_1 is the tagged packet in iteration 1 while it is considered as a disrupting packet in iteration 2). As shown in table.1 the iterations loops (for example in the fifth iteration $\boldsymbol{i_1}$ is again considered as $\boldsymbol{i_{Tag}}$). As soon as two successive iterations for the same $\boldsymbol{i_{Tag}}$ is less than a constant ε (determined by the desired accuracy) we can stop the iteration for that input. The iteration approach provides the latencies of reaching a given output port from all possible input ports of a given node.

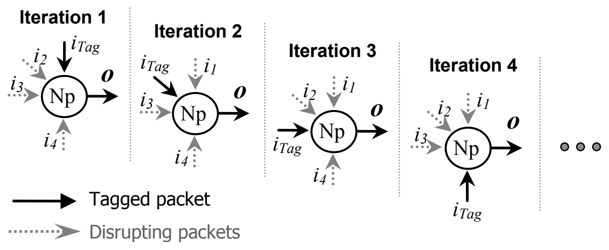

Fig.8 Iterating calculation on different inputs (*i₁, i₂, i₃, i₄*) competing for obtaining the same output (*out*)

Figs 4 and 5 demonstrate the accuracy of the iteration approach for the router N_{15} (the upper left router) in a 5x5 2D-mesh NoC where *only two input ports* south and east compete for the core output port. In iterations 1, 3 and 5 the tagged packet arrives from east and disrupting packet from south and in iterations 2, 4 and 6 is the inverse. For ε=0.01 both mean latencies (*lat-e2c-*$N_{1,5}$ and *lat-s2c-*$N_{1,5}$) converge for a total of 6 iterations.

Table 1. Iterations corresponding to fig 8

Iter num	Tag (i_{Tag})	Disrupting input	Contention probability (*Pr-i2o-*N_p)	MC Result = Computed latency (*lat_*i_{Tag}*2o_*N_p)
1	i_1	i_2	$P_{i2\text{-}to\text{-}o}$ = *F(0)*	*Lat_*i_1*2o_*N_p *(iter 1)*
		i_3	*Pi3-to-o= F(0)*	
		i_4	$P_{i4\text{-}to\text{-}o}$= *F(0)*	
2	i_2	i_1	$P_{i1\text{-}to\text{-}o}$ = *F(Lat_* i_1*2o_*N_p *(iter 1))*	*Lat_*i_2*2o_*N_p *(iter 2)*
		i_3	$P_{i3\text{-}to\text{-}o}$ = *F(0)*	
		i_4	$P_{i4\text{-}to\text{-}o}$ = *F(0)*	
3	i_3	i_1	$P_{i1\text{-}to\text{-}o}$ = *F(Lat_* i_1*2o_*N_p *(iter 1))*	*Lat_*i_3*2o_*N_p *(iter 3)*
		i_2	$P_{i2\text{-}to\text{-}o}$ = *F(Lat_* i_2*2o_*N_p *(iter 2))*	
		i_4	$P_{i4\text{-}to\text{-}o}$ = *F(0)*	
4	i_4	i_1	$P_{i1\text{-}to\text{-}o}$ = *F(Lat_* i_1*2o_*N_p *(iter 1))*	*Lat_*i_4*2o_*N_p *(iter 4)*
		i_2	$P_{i2\text{-}to\text{-}o}$ = *F(Lat_* i_2*2o_*N_p *(iter 2))*	
		i_3	$P_{i3\text{-}to\text{-}o}$ = *F(Lat_* i_3*2o_*N_p *(iter 3))*	
5	i_1	i_2	$P_{i2\text{-}to\text{-}o}$ = *F(Lat_* i_2*2o_*N_p *(iter 2))*	*Lat_*i_1*2o_*N_p *(iter 5)*
		i_3	$P_{i3\text{-}to\text{-}o}$ = *F(Lat_* i_3*2o_*N_p *(iter 3))*	
		i_4	$P_{i4\text{-}to\text{-}o}$ = *F(Lat_* i_4*2o_*N_p *(iter 4))*	
…	…	…	…	…

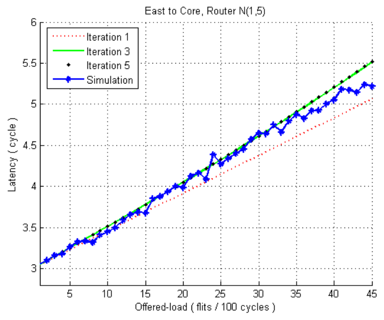

Fig.9 Iterations 1, 3 and 5 for east→core of $N_{1,5}$

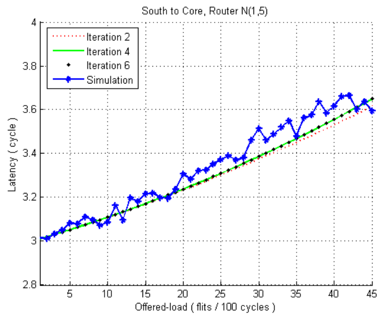

Fig.10 Iterations 2, 4 and 6 for south→core of $N_{1,5}$

6 Experimental results

In order to experimentally evaluate our analytical method we have targeted a NoC architecture arranged in a two dimensional mesh (2D-mesh) topology with an x-first routing algorithm (also called x-y or dimension-ordered) that sends packets first on x then on y axis. X-first algorithm is deterministic, deadlock-free and it guarantees the in-order delivery property of the network [27]. 2D-mesh topology and x-first routing algorithm are adequate choices for regular and homogeneous NoC architecture. DSPIN and ASPIN are two NoCs with such characteristics [8].

Considering an $n*m$ two-dimensional mesh, distribution traffic matrix D has n^2*m^2 elements. According to the uniform traffic pattern each source p sends uniformly the same fraction of its generated load (offered-load) to any of the other $(n*m)-1$ destination q. Also all cores generate data with the same offered-load, so the total load sent from the source p to the destination q i.e. the $(p\ ,\ q)^{th}$ element of distribution matrix D is:

$$d_{pq} = \begin{cases} offered\_load \big/ n*m-1 & p \neq q \\ 0 & p = q \end{cases}$$

For a given router the average rate of data crossing link **i** is:

$$\lambda_i = s_i * offered\_load \,/\, (n*m-1)$$

In this experiment we stay at the network layer, and cores (local subsystems) are considered as sinks (i.e. the cores are always ready to accept the incoming packets with the constant rate of one flit per cycle). Thus the transfer delay to cores is approximated with the mean-value of packet-length. Generating and steady-state analysis of Markov chains has been done by using CADP toolbox [26]. Different paths of a 5x5 2D-mesh NoC were examined for several offered-loads. Latency versus offered-load curves for the paths $N_{5,3}\rightarrow N_{3,5}$, $N_{1,5}\rightarrow N_{5,1}$, $N_{3,1}\rightarrow N_{3,5}$, and $N_{5,3}\rightarrow N_{1,3}$ are presented in figs. 11, 12, 13 and 14 respectively. We remind that these paths are analyzed separately while all nodes send packets to all other nodes according to a uniform pattern. Results obtained by our analytical method are compared with results of a corresponding SystemC CABA simulation platform. Mean packet length for all cores is equal to 10 flits. According to the simulation models router latency (*Dly-router-N*) is 3 cycles. Offered-load is expressed with the average number of flits per 100 cycles.

In the simulations for each path and for each offered-load we have taken the average latency of 1000 received packets. Modeling an infinite source buffer between each traffic generator and the network, indispensable during latency-load measurements [15, 27], enable us to observe the NoC saturation threshold. Without such buffers, generator stops the data injection every now and then and the real offered-load does not match the expected [15]. Once the NoC gets saturated, the source queues and consequently the average latency start growing exponentially. As can be observed, the inaccuracy of our approach (compared with the simulation results) is less than 5% for offered-loads less than 40%. After the offered-load 35% the NoC begins to be saturated and the latency tends to infinity. Our method predicts a saturation threshold of 37% for this NoC architecture.

7 Conclusion

In this paper a novel method based on a simple Markov chain model has been proposed in order to compute the mean latency between two nodes of a given Network-on-Chip. It can determine as well the saturation threshold of the network within a good accuracy (comparing to simulation). Our method has been experimentally evaluated through different paths in a 5x5 2D-mesh NoC and the results has been compared with the results of the corresponding SystemC simulations.

The methodology is almost generic and adaptable to arbitrary NoC topology and deadlock free routing algorithms. It basically addresses deterministic routing algorithms, but also those adaptive algorithms in which the fraction of total data sent through each of the possible paths between a source and a destination can be determined, with the hypothesis that the actual choice remains probabilistic. It works for applications that could be formalized into the form of a distribution traffic matrix, indicating the data rate each core sends to the others. The method covers the output allocation arbitration mechanisms in which the selection between simultaneous packets on the inputs is “fair” in the sense that the arbitration is starvation-free so in the long run it gives equal probability to all inputs for acquiring the shared output. Concerning the high level (end-to-end) flow control, we should mention that it is initially out of the scope of this work since the packet latency is considered from when its header arrives into the network layer until the moment it gets out of the network, while end-to-end flow control is implemented either in network interface or application layers. Although currently we do not cover the case of several VCs, we believe that if the distribution of traffic on each virtual *network* is specified (to have one traffic-matrix and one routing function *per virtual network*) then it is possible to extend our method to several stages of arbitration within each router. A comprehensive study of this possibility is left for future work.

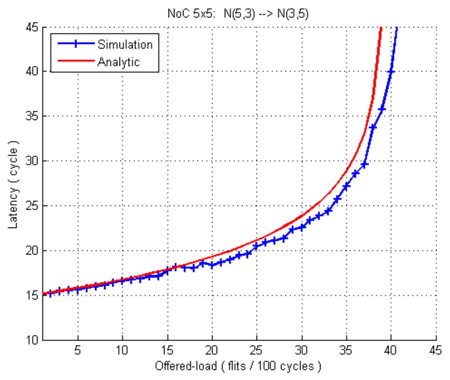

Fig.11 latency/load curve for $N_{5,3} \rightarrow N_{3,5}$

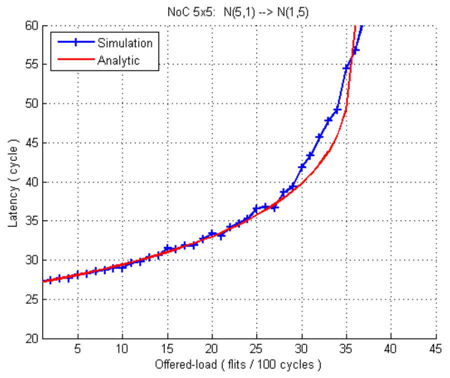

Fig.12 latency/load curve for $N_{5,1} \rightarrow N_{1,5}$

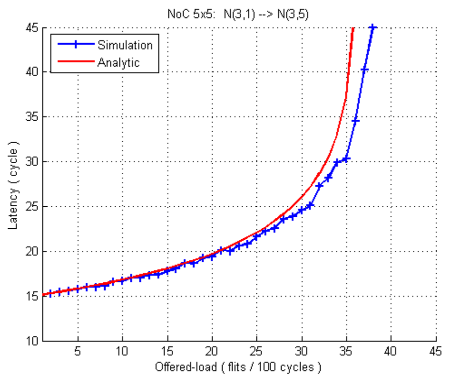

Fig.13 latency/load curve for $N_{3,1} \rightarrow N_{3,5}$

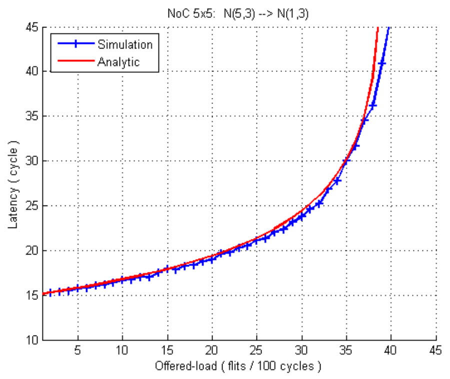

Fig.14 latency/load curve for $N_{5,3} \rightarrow N_{1,3}$

References

[1] A. Jantsch and H. Tenhunen, *Networks on chip*: Kluwer Academic Publishers Boston, 2003.

[2] P. Wielage and K. Goossens, "Networks on Silicon: Blessing or Nightmare?," in *Proceedings of the Euromicro Symposium on Digital Systems Design*: IEEE Computer Society, 2002.

[3] P. Guerrier and A. Greiner, "A generic architecture for on-chip packet-switched interconnections," in *Proceedings of the conference on Design, automation and test in Europe* Paris, France: ACM Press, 2000.

[4] L. Benini and G. De Micheli, "Networks on Chips: A New SoC Paradigm," *Computer,* vol. 35, pp. 70-78, 2002.

[5] K. Goossens, J. Dielissen, and A. Radulescu, "AEthereal Network on Chip: Concepts, Architectures, and Implementations," *IEEE Des. Test,* vol. 22, pp. 414-421, 2005.

[6] T. Bjerregaard, "The MANGO Clockless Network-on-Chip: Concepts and Implementation." vol. PhD thesis: Technical University of Denmark, 2005.

[7] E. Beigne, F. Clermidy, P. Vivet, A. Clouard, and M. Renaudin, "An Asynchronous NOC Architecture Providing Low Latency Service and Its Multi-Level Design Framework," in *Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems*: IEEE Computer Society, 2005.

[8] A. Sheibanyrad, A. Greiner, and I. Miro Panades, "Multisynchronous and Fully Asynchronous NoCs for GALS Architectures," *IEEE Design & Test of Computers,* vol. 25, pp. 572-580, 2008.

[9] M. Coppola, M. D. Grammatikakis, R. Locatelli, G. Maruccia, and L. Pieralisi, *Design of Cost-Efficient Interconnect Processing Units: Spidergon STNoC*, 2008.

[10] S. Manolache, "Schedulability analysis of real-time systems with stochastic task execution times," 2002.

[11] J. Kim and K. G. Shin, "Execution time analysis of communicating tasks in distributedsystems," *Computers, IEEE Transactions on,* vol. 45, pp. 572-579, 1996.

[12] I. Cohen, O. Rottenstreich, and I. Keslassy, "Statistical approach to NoC design (extended version)," Technical Report TR08.

[13] V. Soteriou, H. Wang, and L. Peh, "A Statistical Traffic Model for On-Chip Interconnection Networks," 2006, pp. 104-116.

[14] G. V. Varatkar and R. Marculescu, "On-chip traffic modeling and synthesis for MPEG-2 video applications," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on,* vol. 12, pp. 108-119, 2004.

[15] E. Salminen, A. Kulmala, and T. D. Hamalainen, "On network-on-chip comparison," 2007, pp. 503-510.

[16] L. Benini, D. Bertozzi, D. Bruni, N. Drago, F. Fummi, and M. Poncino, "Legacy SystemC co-simulation of multi-processor systems-on-chip," 2002, pp. 494–499.

[17] S. G. Pestana, E. Rijpkema, A. Radulescu, K. Goossens, and O. P. Gangwal, "Cost-Performance Trade-Offs in Networks on Chip: A Simulation-Based Approach," in *Proceedings of the conference on Design, automation and test in Europe - Volume 2*: IEEE Computer Society, 2004.

[18] A. Sheibanyrad, I. Miro Panades, and A. Greiner, "Systematic comparison between the asynchronous and the multi-synchronous implementations of a network on chip architecture," in *Design Automation and Test in Europe (DATE)* Nice, France: EDA Consortium (IEEE, ACM), 2007, pp. 1090-1095.

[19] A. E. Kiasari, D. Rahmati, H. Sarbazi-Azad, and S. Hessabi, "A Markovian Performance Model for Networks-on-Chip," 2008, pp. 157-164.

[20] P. Hu and L. Kleinrock, "An analytical model for wormhole routing with finite size input buffers," 1997, pp. 549-560.

[21] Z. Guz, I. Walter, E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "Network Delays and Link Capacities in Application-Specific Wormhole NoCs," *Special Issue of the Journal of VLSI Design,* 2007.

[22] A. Khonsari, M. Ould-Khaoua, and J. Ferguson, "A General Analytical Model of Adaptive Wormhole Routing in k-Ary n-Cube Interconnection Networks," *SIMULATION SERIES,* vol. 35, pp. 547-554, 2003.

[23] W. J. Dally, "Performance analysis of k-ary n-cube interconnection networks," *IEEE Transactions on Computers,* vol. 39, pp. 775-785, 1990.

[24] H. Sarbazi-Azad, A. Khonsari, and M. Ould-Khaoua, "Analysis of deterministic routing in k-ary n-cubes with virtual channels," *Journal of Interconnection Networks,* vol. 3, pp. 85-101, 2002.

[25] U. Y. Ogras and R. Marculescu, "Analytical router modeling for networks-on-chip performance analysis," 2007, pp. 1-6.

[26] H. Garavel, F. Lang, and R. Mateescu, "An overview of CADP 2001," *European Association for Software Science and Technology (EASST) Newsletter,* vol. 4, pp. 13-24, 2002.

[27] W. J. Dally, *Principles and practices of interconnection networks*: Morgan Kaufmann, 2004.